SpaceWire 2016

7th International Conference
25th-27th of October 2016
Yokohama, Japan

2016.spacewire-conference.org

JAXA esa NASA ROSCOSMOS Orchestrating a brighter world NEC MITSUBISHI ELECTRIC Changes for the Better MITSUBISHI HEAVY INDUSTRIES, LTD. OSAKA UNIVERSITY UNIVERSITY OF DUNDEE

SpaceWire-2016

Proceedings of the 7th International SpaceWire Conference

Yokohama 2016

Editors: Steve Parkes and Carole Carrie

Space
Technology
Centre
University of Dundee

SpaceWire-2016

Proceedings of International SpaceWire Conference

Yokohama 2016

ISBN: 978-0-9954530-0-5

Space Technology Centre
University of Dundee

© **Space Technology Centre**
University of Dundee
Dundee
2016

All rights reserved. No part of this publication may be reproduced or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Preface

These proceedings contain the papers presented at the 2016 International SpaceWire Conference, held in Yokohama, Japan between 24th and 27th October, 2016. The International SpaceWire Conference brings together international spacecraft engineers and academics who are working on spacecraft on-board data-handling technology. It is of benefit to product designers, hardware engineers, software engineers, system developers and mission specialists interested in and working with SpaceWire, enabling them to share the latest ideas and developments related to SpaceWire and SpaceFibre technologies.

SpaceWire is now being used or designed into well over one hundred spacecraft, covering science, exploration, Earth observation and commercial applications. High profile missions like James Webb Space Telescope, GAIA, ASTRO-H, ExoMars, Bepicolombo, Sentinels 1, 2, 3 and 5 precursor, and GOES-R are using SpaceWire extensively. SpaceWire is being used in Europe, Japan, USA, Russia, China, India, and other countries of the World.

SpaceFibre is the next generation of SpaceWire technology, offering higher data-rates and substantially enhancing the capabilities of SpaceWire. It runs over electrical or fibre optic cable covering distances of 5m and 100 m respectively while running at data rates of up to 3.125 Gbits/s with 6.25 Gbits/s currently under development. SpaceFibre incorporates quality of service, providing multiple independent virtual networks for transferring information over the physical network, each virtual network having its own priority, bandwidth allocation and schedule. These capabilities enable SpaceFibre to provide deterministic data delivery without loss of network bandwidth for combined control and payload data-handling networks. It also provides integrated, rapid fault detection, isolation and recovery technology, which makes SpaceFibre a highly robust network for use in applications where reliability and availability are critical. SpaceFibre multi-laning technology extends the bandwidth of a link using multiple lanes and provides hot and cold redundancy and graceful degradation. Asymmetric links are also possible, using uni-directional lanes provided at least one lane is bi-directional. SpaceFibre uses the same packet format and addressing concepts as SpaceWire making it trivial to connect existing SpaceWire equipment into a SpaceFibre network. IP cores for radiation tolerant FPGAs and ASICs with SpaceFibre interfaces are available and under development. A wide range of test and development equipment is now available.

The conference covers many different aspects of SpaceWire and SpaceFibre technology and includes both academic and industrial presentations. Sessions address recent developments of the SpaceWire set of standards, space missions and other applications using SpaceWire, new components, sensors and cables which support the SpaceWire standard; products supporting SpaceWire including onboard equipment, instruments and related onboard software; methods and equipment to aid the test and verification of SpaceWire components, units and systems; and SpaceWire networks, their architecture, configuration, and discovery, as well as higher level protocols and related hardware and software design issues. The new sessions on SpaceFibre illustrate how this next generation of SpaceWire technology is gaining momentum, already being designed into spaceflight systems. It is an exciting time in the SpaceWire community as this latest technology literally begins to take off.

Technical seminars at the conference cover SpaceWire-D which provides deterministic data deliver using existing SpaceWire devices and SpaceFibre.

The community of engineers working on SpaceWire meet regularly at the SpaceWire Working Group meetings to help with the further development of SpaceWire, SpaceFibre and related standards and technologies. This group includes engineers from many parts of the World with substantial contributions from Europe, USA, Japan, and Russia. The SpaceWire Conference complements these Working Group meetings with more formal presentations from a wider range of contributors.

The conference committee would like to acknowledge the support and hard work of the many individuals who made International SpaceWire Conference 2016 a reality. First, we thank the authors and the keynote speakers for their high-quality contributions. We express our gratitude to the Technical Committee for their assistance in the review process. We thank all people supporting us at JAXA, the Space Technology Centre, University of Dundee, and the European Space Agency. Finally, we would like to give a special thanks to Carole Carrie at STAR-Dundee for making the conference happen.

The Conference Chairpersons,

Masaharu Nomachi, *Osaka University, Japan*
Steve Parkes, *Space Technology Centre, University of Dundee, UK*
Dirk Thurness, *European Space Agency, The Netherlands*

Technical Committee

Brice Dellandrea – Thales Alenia, France

Seisuke Fukuda – JAXA/ISAS, Japan

Wahida Gasti - ESA, The Netherlands

Sandi Habinc – Cobham Gaisler, Sweden

Hiroki Hihara – NEC, Japan

Christophe Honvault – ESA, The Netherlands

Torbjörn Hult - RUAG Space, Sweden

Jørgen Ilstad - ESA, The Netherlands

Paul Jaffe - Naval Research Laboratory, USA

David Jameux- ESA, The Netherlands

Clifford Kimmery – Honeywell Inc., USA

Alexander Kisin - MEI, USA

Robert Klar - South West Research Institute, USA

Jim Lux - NASA JPL, USA

Keiichi Matsuzaki – JAXA, Japan

Giuseppe Montano – Airbus, UK

Minoru Nakamura – Mitsubishi Electric, Japan

Masaharu Nomachi – University of Osaka, Japan

Olivier Notebaert - Astrium SAS, France

Steve Parkes - University of Dundee, Scotland, UK

Paul Rastetter - Astrium GmbH, Germany

Alan Senior – Thales Alenia Space UK Ltd, UK

Yuriy Sheynin - St. Petersburg State University of Aerospace Instrumentation, Russia

Tatiana Solokhina - ELVEES, Russia

Martin Suess - ESA, The Netherlands

Antonis Tavoularis – Teletel, Greece

Dirk Thurnes – ESA, The Netherlands

Raffaele Vitulli - ESA, The Netherlands

Takahiro Yamada - JAXA/ISAS, Japan

Programme Overview

Monday 24 October

14:00 – 17:30 Registration

14:00 – 17:30 Tutorials of SpaceFibre and SpaceWire-D

Tuesday 25 October

09:15 – 17:00 Registration

09:30 – 10:35 Conference Opening / Keynote Presentations (65 min)

10:35 – 11:05 Missions & Applications Short (30 min)

11:35 – 12:50 SpaceFibre 1 Long (75 min)

14:05 – 15:20 Networks & Protocols Short (75 min)

15:45 – 17:30 Components Short (105 min)

Wednesday 26 October

09:15 – 11:30 Registration

09:15 – 10:55 Missions & Applications Long (100 min)

11:30 – 11:45 Test & Verification Short (15 min)

11:45 – 12:30 SpaceFibre Short (45 min)

13:45 – 15:50 Networks & Protocols 1 Long (125 min)

15:50 – 17:00 Poster Session (70 min)

Thursday 27 October

09.15 – 11:00 Registration

09:15 – 10:30 Components Long (75 min)

11:00 – 12:40 Networks & Protocols 2 Long (100 min)

13:55– 15:10 SpaceFibre 2 Long (75 min)

15:40– 16:55 Test & Verification Long (75 min)

Programme is subject to change

Tuesday 25 October

Missions & Applications (Short)

Using SpaceWire Time Codes for Spacecraft Time Synchronization

SpaceWire Missions and Applications, Short Paper

Susan C. Clancy, Mazen M. Shihabi, Krisjani S. Angkasa
Flight Communications Systems Section
Jet Propulsion Laboratory
Pasadena, California 91109 USA
Susan.Clancy@jpl.nasa.gov, Mazen.M.Shihabi@jpl.nasa.gov, Krisjani.S.Angkasa@jpl.nasa.gov

Abstract— This paper describes how SpaceWire Time Codes can be used for synchronizing time within various subsystems of a spacecraft as well as, maintaining a common time reference needed for coordinating operations within a spacecraft. The algorithms to account for inaccuracies in the time distribution method were based on the NASA-4009 Space Telecommunication Radio System (STRS) standard [1], which defined an interface for synchronizing clocks running at different tick rates and tick resolutions.

***Index Terms*— Relevant indexing terms: SpaceWire, SpaceWire Time Codes, SpaceWire Time Distribution Protocol, CCSDS Unsegmented Time (CUC), Space Telecommunications Radio System (STRS).**

I. Introduction

Spacecraft systems are typically comprised of many subsystems, each with their own clock running at different tick rates and with varying performance, which can degrade over time. Clock synchronization becomes very important in cases where commands and activities need to be correlated with a common time reference and for attitude determination based on current time or predicted position propagated over a period of time.

Subsystems needs to know what time it is in order to perform synchronized activities, or to time-tag telemetry that can be correlated with operations in other subsystems. One subsystem equipped with a Ground Navigation Satellite System (GNSS) receiver can maintain an accurate reference of time and can act as the time "master" to distribute the time to other nodes connected via SpaceWire.

II. Spacecraft Time Synchronization Methods

There are two common methods used for synchronizing time on a spacecraft: (1) a periodic "message" based method performed in software and (2) a periodic "hardware tick" based method performed in hardware or firmware.

The "message" based method uses a "master" to generate a "tick" message at specific intervals and sends a time message to the "slaves" at a specific "tick". The "slaves" update their time at a time boundary after the time message is received. In the example below, the "tick" message is sent 100 times per second, and the time message is sent once per second prior to the one second time boundary.

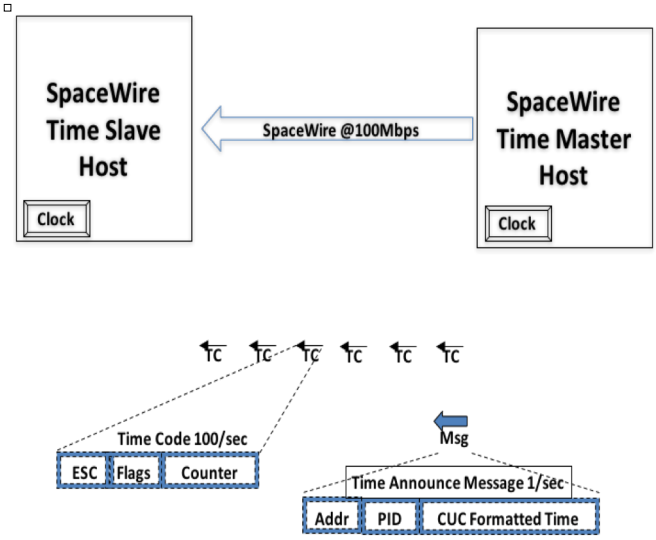


Fig. 1. Time Synchronization "Message" Based Method

The "hardware tick" method uses a "master" to send a "tick" signal to all the "slaves", who will then increment their own slave clock. The hardware clock oscillator used to generate the clock tick signal is usually a Temperature Compensated Crystal Oscillator (TCXO) or Ovenized Crystal Oscillator (OCXO) with accuracy better than 1 part per million.

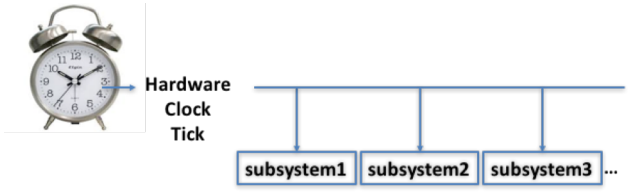


Fig. 2. Time Synchronization "Hardware Tick" Method

III. Spacecraft Time Synchronization Challenges

The challenges in synchronizing spacecraft time are similar to those in ground-based systems:

A. Latency – the time it takes to transfer and respond to a time update. Each spacecraft subsystem must account for latency and be tolerant within a measured minimum and maximum range. A technique for measuring latency is described in the SpaceWire Time Distribution Protocol [2].

B. Jitter – the intermittent delay in the path between the master sending the time and the slave receiving and updating their time. Each spacecraft subsystem must tolerate a measured maximum jitter.

C. Drift – the variation in the clock tick rate due to oscillator performance, which typically degrades over time and varies with temperature. The time "master" clock must be calibrated periodically to account for the drift in the time conversion. The drift can be accounted for as a clock rate correction [2] to mimic the actual clock rate changes.

D. Time conversion – the different clocks may tick at different rates and a conversion from the hardware clock value to the time representation unit (usually in seconds) is applied using the clock tick rate, clock hardware value, and an offset, which typically includes drift. The conversion algorithm needs to account for latency, varying jitter, and clock degradation.

A further complication is that the performance of the clock oscillators in various parts of the system may be orders of magnitude different: a spacecraft computer may have a clock with 10 ppm performance, while spacecraft radios and GNSS receivers may be accurate to parts per billion (ppb). The system design, however, may be that all systems need to follow the time kept by the spacecraft computer, so the time distribution method must allow a better clock to follow a poorer clock, which is different than the typical Network Time Protocol (NTP) architecture, where clocks at a lower stratum follow more accurate clocks at a higher stratum.

IV. STRS Time Synchronization Method

The NASA-STD-4009 Space Telecommunications Radio System (STRS) architecture standard [1] defines some time related functions and corresponding Application Programming Interfaces (APIs) for getting, setting, and synchronizing time. These functions are used by applications to maintain and coordinate time derived from different clocks that may have different tick rates and resolutions.

Note that the reference clock may or may not have a higher performance and stability than the monitored clock. The purpose is to synchronize the clocks and not to maintain the correct time. The reference clock and managed clock can exist on the same local host or on different hosts but can be synchronized to report the same time.

The core concept of the STRS clock model is that the underlying clock is allowed to run unhampered, and the relationship between the raw clock and "time" is encapsulated in the API which provides a standardized way of getting and setting time based on calling API functions that can account for latency, jitter, and drift using conversion data. This conversion data is set to values that initially synchronize the reference clock with the managed clock. The conversion data can be updated periodically to continuously account for drift.

The linear conversion algorithm commonly used to compute time, converts hardware clock ticks to time in seconds using the oscillator clock rate and hardware clock ticks as follows:

$$time = (clock\_rate \times clock\_ticks) + offset$$

The STRS time conversion algorithms include additional adjustments to the rate and offset to account for the difference between two clocks plus the latency, drift, and even jitter as follows:

$$STRS\ time = ((clock\_rate + adjust\_rate) \times clock\_ticks) + (offset + adjust\_offset)$$

Figure 3 below shows an implementation of an STRS time interface that synchronizes a local reference clock and a local managed clock. The conversion data is applied when getting the time via the STRS_GetTime API function which converts the clock value to a time in seconds and sub seconds.

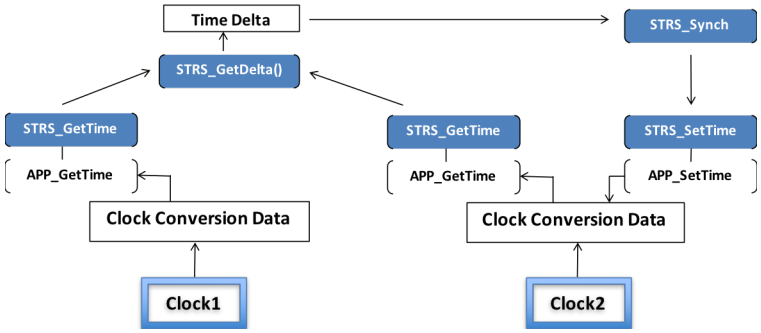


Fig. 3. STRS Time Synchronization Method

V. SpaceWire Specification for Time Codes

The SpaceWire Protocol Standard [3] includes the definition of the time interface with Time Codes and the TickIn and TickOut signals. The key features in any implementation are:

A. Time Code generation or receipt can be enabled or disabled.
B. The Time Code rate is generated by a "master" and can be configured to send Time Codes at a specific rate
C. The Time Code is a specific type of SpaceWire message containing a Time Code identifier and Time Code counter. The Time Code counter is an incrementing 0 to 63 integer value and any missing Time Code can be detected and reported by firmware using this counter.

The Time Code TickIn / TickOut signals can support an interface to a software interrupt line and/or hardware signal going to a hardware clock. The time "master" (aka initiator) can generate a software interrupt for each tick using the TickIn signal. Using this TickIn interrupt, a "slave" (aka target) can implement a SpaceWire "derived clock" to align the tick generation with the time message.

VI. CCSDS Time Message Format

The CCSDS Unsegmented Code (CUC) Time Specification [4] is a proposed standard for specifying time as a number of seconds and sub-seconds.

CCSDS Unsegmented Code			
P-Field		T-Field	
1st	2nd	CoarseTime	FineTime

Fig. 4. CCSDS Unsegmented Code (CUC) Format

The fields in the time announced message are as follows:

Unsegmented Code			
P-Field		T-Field	
Addr	0xF1	32-bit time (sec)	16-bit (not used)

Fig. 5. Time Announced Message Format

VII. Time Synchronization Demo

The first goal was to demonstrate the ability to compensate for time distribution inaccuracies due to latency, jitter, and drift using the STRS time API. The second goal was to demonstrate time distribution using SpaceWire Time Codes and the CCSDS CUC formatted time message.

In the first test, the time synchronization was performed on the SDR using the Clock Calibration waveform component (CLKCAL) to synchronize two different clock "kinds" on the SDR. The CLKCAL waveform (1) computes the delta between the reference clock time and managed clock time, (2) computes the drift detection value for each clock, (3) reports any time delta or drift detection, and (4) synchronizes the managed clock to report the same time as the reference clock. The STRS time API is used by CLKCAL for getting, setting, and synchronizing the time.

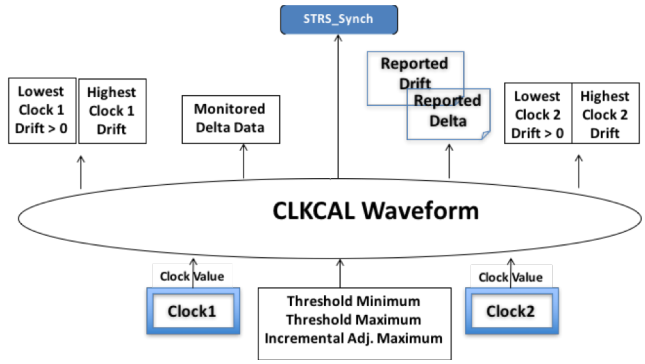


Fig. 6. Clock Synchronization Test

In the second test configuration, the CLKCAL waveform was integrated with the SpaceWire Time interface. The SpaceWire time interface on the SDR "slave" was implemented as a "waveform" component with counterparts running in both firmware on a Field Programmable Gate Array (FPGA) and software running on the SDR Sparc computer.

The SPW waveform continuously (1) receives the time codes, (2) maintains a Time Code tick counter, (3) captures the time sent in the SpaceWire time messages, (4) sends periodic notifications at synchronization intervals and (5) makes the time available to other waveforms.

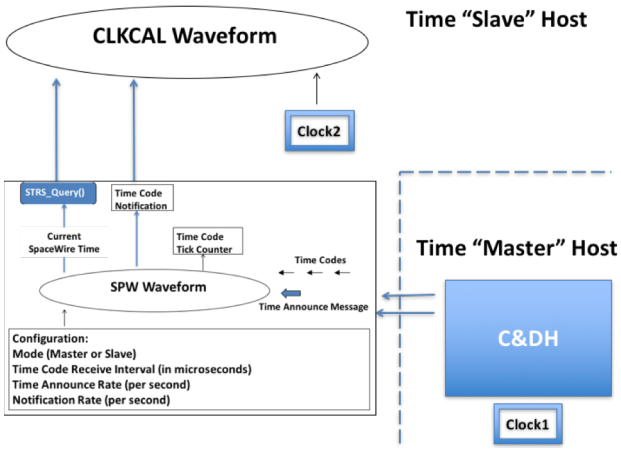


Fig. 7. Clock Synchronization with SpaceWire Test

The time delta is computed by CLKCAL and is expected to be constant unless inaccuracies are introduced by jitter or drift.

A set of "threshold" values (minimum, maximum, and rate adjustment maximum) is used to determine when to synchronize the clocks and which method to use (time jump,, incremental update, or a rate adjustment).

The threshold minimum accounts for expected jitter introduced by the time distribution interface itself. The

minimum should not be 0 since there will always be some amount of jitter. The threshold minimum value can be determined by analyzing the delta values over a period of time.

Any delta above the threshold minimum but below the rate adjustment maximum will cause a rate adjustment update to synchronize the clocks. The rate adjustment is included in the conversion data used in the time conversion algorithm. This is the smoothest update method. Any delta between the minimum and the incremental adjustment maximum will use an incremental adjustment over a period of time. Incremental updates will be made until they add up to the desired delta. This adjustment period can be longer to make smaller incremental updates or shorter to make bigger incremental updates. Any delta above the incremental adjustment maximum will cause a time "jump". A "jump" is not desired when the managed clock is used for time based computations or activities but is a common method used for updating or synchronizing time during initialization.

The clock drift is obtained by capturing a counter for each clock at specific intervals. This counter should remain constant unless the clock is drifting. Watermarks are used to track the range of drift for each of the clocks. A drift watermark reporting threshold maximum value is used to determine when to report drift. This reporting threshold can be 0 to always report any detected drift or a value that must be exceeded before the drift is reported.

VIII. TEST RESULTS

The initial tests run on the SDR show the STRS time interface successfully synchronizing two different clock "kinds" that exist on the same SDR. The data below (in red) shows the software detecting the delta above the threshold, and performing the synchronization.

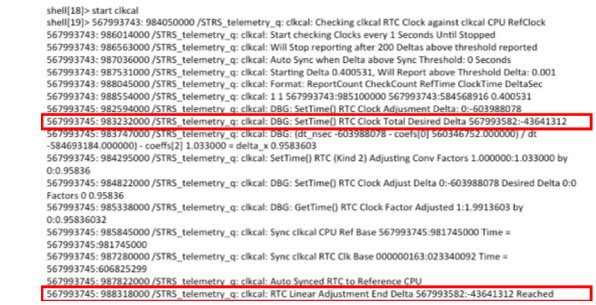


Fig. 8. Clock Synchronization

The clock delta and drift reported by the CLKCAL waveform used inputs distorted by the jitter introduced by the software itself due to running in a multitasking environment on both the "master" and "slave". This artificial input data was useful in developing and testing the clock synchronization thresholds and synchronization response. The use of an independently generated counter latched at fixed intervals as described in earlier work in [2] and a "distributed" interrupt generated via the TickOut signal as described in [6] are needed to account for the real inaccuracies introduced by latency, jitter, and drift.

The synchronization parameters that were tested included thresholds to control whether time was updated gradually or immediately in one-time jump.

The following test result shows the "threshold min." should be set to 6 usec to avoid synchronization for changes smaller than the expected. 1 to 5 usec range. Based on this example, the changes above 5 usec would result in a clock synchronization.

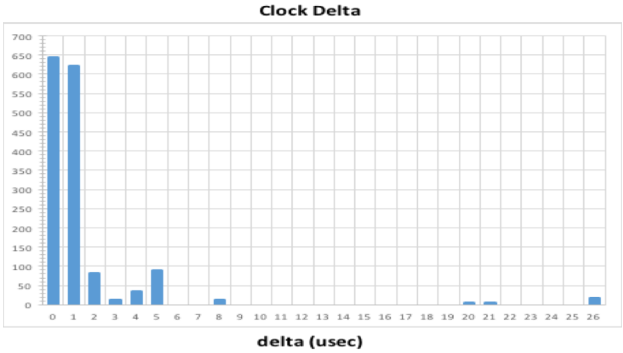


Fig. 9. Synchronization Delta Values

In earlier tests on the SDR, the CLKCAL waveform attempted to poll the received Time Code counter value to increment the Time Code virtual clock ticks. These tests intermittently failed when generating Time Codes at 100 per second. The "slave" reported a missed tick error when the Time Code value did not increment as expected, although this issue was not encountered when Time Codes were generated at once per second.

The TickOut interrupt interface and a latched counter interface have since been implemented in the SDR FPGA firmware to mitigate these issues. The TickOut interrupt unit tests showed that software increments the SpaceWire DCLK virtual ticks properly. However, tests using these mechanisms integrated with CLKCAL are planned for the future.

IX. CONCLUSIONS AND FUTURE WORK

The STRS time API does accommodate synchronizing various clock "kinds" using clock compensation data to mitigate inaccuracies (latency, jitter, drift) in a time distribution system.

Synchronization tolerance ranges (i.e. thresholds) can be used to determine which method to use for synchronizing clocks and when to correct for drift. Future work is needed to establish the tolerance ranges for synchronizing clocks using the SpaceWire Time Distribution Protocol such as those described in [2] and [6].

The SpaceWire Time Codes are useful for creating a virtual clock on hosts connected via SpaceWire. This SpaceWire virtual clock can be implemented on a "slave" host that may not have a clock.

ACKNOWLEDGEMENT

The authors wish to acknowledge the following individuals for their significant contributions: James P. Lux, Minh Lang and David E. Robison from the Jet Propulsion Laboratory, and David Chelmins and Larry M. Vincent from the NASA Glenn Research Center.

This work was carried out at the Jet Propulsion Laboratory in Pasadena (JPL), California, under contract with the National Aeronautics and Space Administration, for the SCaN Testbed Project. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the U.S. Government or the Jet Propulsion Laboratory.

©2016 California Institute of Technology. Government sponsorship acknowledged.

REFERENCES

[1] NASA-STD-4009 Space Telecommunication Radio System (STRS) architecture standard

[2] A. Sakthivel, J. Ekergarn, D. Hellström, S. Habinc, M. Suess, "SpaceWire Time Distribution Protocol Implementation and Results," 6th International SpaceWire Conference, [September 2014]

[3] ECSS-E-ST-50-12A-C SpaceWire Protocol Standard

[4] CCSDS Time Code Format Recommended Standard, Issue 4 CCSDS-301.0-B-4 Blue Book November 2010

[5] Space Communications and Networking (SCaN) Test Bed Experiment https://spaceflightsystems.grc.nasa.gov/sopo/scsmo/scan-testbed/

[6] S. Habinc, A. Sakthivel, M. Suess, "SpaceWire – Time Distribution Protocol," International SpaceWire Conference, [June 2013].

Maturation of a Scalable Form Factor System Standard for Interoperable Spaceborne Processing and Interconnect Needs

Missions and Applications, Short Paper

Joseph R. Marshall, Richard W. Berger
BAE Systems
Manassas, VA 20112
joe.marshall@baesystems.com

***Abstract*— This paper will briefly review the SpaceVPX standard with special emphasis on the interconnect planes between the modules. Comparisons to other form factor standards will be included. SpaceWire and its usage as the control plane in a SpaceVPX system or as a medium speed data plane will be discussed. A summary and status of any updates or future efforts involving the SpaceVPX standard will be included.[1]**

***Index Terms*—Standards, SpaceWire, networking, spacecraft electronics, SpaceVPX, MicroTCA, CompactPCI, PC/104, RapidIO, form factor, fault tolerance, redundancy**

I. INTRODUCTION

Future spaceborne systems will require additional onboard processing and much greater interface connectivity. Many efforts worldwide are starting to address these needs. SpaceVPX, a recently released ANSI/VITA standard, was created to provide the structure and definition for interoperable modules that will be created to meet these needs. It provides a multi-layer set of fabrics using serializer/deserializer (SERDES), LVDS and LVCMOS devices to provide interconnections in a scalable and fault tolerant way. Initial fabrics used by SpaceVPX are RapidIO, SpaceWire and I2C. Provisions are provided for heritage or user defined interfaces to interact with these within the structure. SpaceWire is setup as both a control plane for command and data handling throughout the box as well as a medium speed data plane. SpaceVPX was approved and released by ANSI and VITA in April 2015 as VITA 78. Since then, multiple organizations are utilizing it to create interoperable modules.

Building on previous SpaceWire network elements, BAE Systems is creating a set of silicon application specific standard products (ASSP) [1] [2] [3] to provide power efficient general purpose building blocks for the creation of scalable SpaceVPX modules across these three fabrics. These building blocks are key to a new family of SpaceVPX processing and network modules [4] being developed for a wide variety of space applications. One of the advantages of using SpaceVPX is the significant industry heritage of OpenVPX modules, backplanes, chassis, power supplies and test equipment. Exploring optimal methods for leveraging these elements is an important part of the development of the BAE Systems SpaceVPX modules.

The SpaceVPX working group is monitoring the usage of the standard identifying potential upgrades and enhancements. For instance, SpaceVPXLite (VITA 78.1) will focus on building more limited and smaller 3U sized systems of smaller number of slots yet maintaining the full fault tolerance of the parent standard.

II. FORM FACTOR STANDARDS

In the non-space world there are several form factor standards that are being used for high performance heterogeneous systems. A summary of a cross section of these is captured in Table I. Of these CompactPCI [5], SpaceVPX [6], MicroTCA [7] and PC/104 [8] have been applied to spaceborne applications. One of the key challenges of larger systems is their fault tolerance. Some of these standards were fully designed to provide a basis for a single point fault tolerant system. Other standards contain redundancy provisions that may provide some fault tolerance (e.g. multiple fabrics that may be used between modules) yet also included features (e.g. common power feeds or busses) that formed single points of failure for the system. The last class, usually representing standards for small numbers of modules, have little or no redundancy and make the assumption that redundancy will be provided at the box level, switching out an entire box if the primary fails to provide the services required of it. SpaceVPX built on the other existing standards at the time and made adjustments and changes so that it could fully support robust single point fault tolerance or more across its system implementations. This is done mostly through radial or star distribution of most interfaces and resources within the box.

As Table I shows, there are different amounts and types of profiles across form factor standards. Some, like CompactPCI provided minimum flexibility just defining a handful of slot profiles, while others, such as OpenVPX and by extension, SpaceVPX provided users with many profiles in slots, modules, backplanes and chassis.

SpaceVPX is the only current standard in the table that supports the usage of SpaceWire. Some OpenVPX slots

[1] Approved for Public Release – ES-ISR-082316-0109

contain control plane wiring for PCI Express that are compatible with SpaceWire routings.

TABLE I. FORM FACTOR STANDARDS

Standard	# Pins	Interfaces	Fault Tolerance	Physical	Backplane Voltages	Profiles
CompactPCI PICMG 2.3 [9]	3U: 132 / 6U: 264	PCI, I2C, JTAG, user defined	Single bus	3U-160 or 6U-160 by 0.8" Air / Conduction Cooled	3.3V and 5V, 12V, -12V	Bus Clock, Width, Slots
CompactPCI Serial [10]	3U: 538 6U: 600	PCI, PCIe, SATA/SAS, USB 2.0/3.0, Ethernet, I2C, JTAG, user defined	Single Bus, multiple fabrics	3U-160 or 6U-160 by 0.8" Air / Conduction Cooled	3.3V and 5V, 12V, -12V, 48V	Slots
OpenVPX VITA 65 [11], 46 [12], 48.2 [13]	3U: 320 / 6U: 832	RapidIO, PCIe, Ethernet, I2C, JTAG, RF, Optical, user defined	Supports 1 of 2 to M of N common power and clocks	3U-160 or 6U-160 by 0.8", 0.85" and 1.0" Air / Conduction Cooled	12V, 5V and 3.3V, -12V	Slot, Module, Backplane, Chassis
SpaceVPX VITA 78 [14]	**3U: 320 / 6U: 832**	**RapidIO, Ethernet, SpaceWire, PCI, I2C, JTAG, user defined**	**Supports 1 of 2 to M of N No single point of failure**	**3U/6U -160/220/280/340 by 0.8", 1.0" or 1.2" Air / Conduction Cooled**	**12V, 5V and 3.3V, -12V**	**Slot, Module, Backplane, Chassis**
MicroTCA [15]	B: 85 B+/AB: 170 A+B: 340	RapidIO, Ethernet, PCI Express, Fibre Channel, I2C, JTAG, user defined	Supports 1 of 2 to M of N No single point of failure	74/149 x 180mm x 3 heights Air / Conduction Cooling	+12 and 3.3V	size / protocol, MCH types
VITA 78.1 (in development)	3U: 320	RapidIO, Ethernet, SpaceWire, I2C, JTAG, RF, Optical, user defined	Supports 1 of 2 to M of N No single point of failure	3U-160/220/280/340 By 0.8", 1.0" or 1.2"	12V, 5V and 3.3V, -12V	Slot, Module, Backplane, Chassis
PC/104 Family [16]	Top: 120 Bottom: 156	PCI, PCI Express, USB 2.0/, SM Bus, SATA, LPC	Single string; multiple fabrics	3.55" by 3.775" stacked EPIC 4.528"x6.496" EBX 5.75" x 8"	3.3V, 5V, 12V, -12V	Module Types

III. HISTORY

In 2011, a group of industry experts and government officials met as part of the GOMACTech conference. They discussed how the space industry would soon require more processing and data bandwidth onboard than the typical spaceborne CompactPCI box could provide. SpaceWire was already a popular fabric with the capability of exceeding the 1 or 2 Gbps bandwidth that a CompactPCI box could provide to share between the modules within. This success with SpaceWire, low speed 1 Gbps SERDES links and the growing differences between COTS systems which now used SERDES-based fabrics and space systems pointed toward SERDES based fabrics for space. Radiation hardened or tolerant technology was emerging that could support higher performance SERDES and thus higher internal bandwidths. Due to the high cost of development, there was general feeling that these new high speed interfaces be standardized across the space community. The Next Generation Space Interconnect Standards group was formed at that meeting and has since focused on its selected high performance interface, RapidIO. Within a year of its formation and following a successful set of trade studies and use case analysis that arrived at a consensus to focus on RapidIO, SpaceWire and I2C as a three tiered set of interconnect fabrics, the NGSIS group realized it also needed a form factor standard for physical implementations of the interfaces that could produce interoperable modules. Once again, the consensus of the group settled on the OpenVPX standard as the best base to build upon. [17] [18] [19]

Three years later, after over 50 drafts, hundreds of telecons and face to face meetings with contributions and reviews from across the space industry, VITA 78.00 was ratified by VITA and ANSI in April 2015. This 400+ page standard was built strongly on OpenVPX so that it would be possible to use the less expensive OpenVPX modules and chassis for prototyping SpaceVPX systems, for driving SpaceVPX modules with the various fabrics and for testing SpaceVPX modules in an existing infrastructure adapted for SpaceVPX modules.

IV. SPACEVPX STANDARDIZATION

SpaceVPX provides multiple levels of standardization for space electronics modules. First, it defines a common connector and backplane structure that has been tested for use in many high vibration (e.g. ships and aircraft) environments with much longer durations then typical spacecraft needs. Three variations of the connector are available from three different manufacturers so the best one may be picked for a specific box. Although the connectors are not intermate-able, their footprint is such that modules may be changed from one to the other without a printed wiring board update.

The connector is divided into segments and multiple profiles are defined that map the many pins to interface planes in identical locations. Profiles are defined to provide basic functions such as switches, controllers, payloads or peripherals. These planes are then mapped at the module level to hold specific protocols and speed selections that are compatible with those pin layouts. This mapping is further described in the next section. Any user defined pins in a profile may be used for any usage. However, it is strongly recommended that the usage is not interfering or can be disabled or not populated if the module is targeted for reuse in other systems not requiring the user defined purposes.

The next level of standardization is provided to handle fault tolerant switching of the utility plane. The utility plane is used for handling the basic operation of a SpaceVPX box. One of at least two controllers direct this operation and are the first logic modules to receive power. The master controller then decides which modules it needs powered to complete a mission. The Utility plane provides power, resets, common low skew clocks and a system management (SM) interface for the controller to interact with utility plane switches and the controlled logic modules. I2C, enhanced with a reset and error status signal, forms the SM interface. In OpenVPX this is bussed between modules in most implementations. In SpaceVPX, it was decided that a star or radial distribution provided much more fault tolerance and potential error containment. The switching of the utility signals and the power to the logic modules is implemented in a Space Utility Management (SpaceUM) module. From a reliability point of view, it is a separate physical module that is actually an extension of the controller for redriving the control signals, an extension of the power supply for redriving the power busses and an extension of each module containing a power switch for each supplied voltage and a control signal switch or selection for providing a single set of control signals to each module. Through the use of fault containment regions in the SpaceUM module, there are no common points of failure that cannot be allocated and controlled back to a power supply, controller or logic module. Thus no module level redundancy is required for a SpaceUM module. SpaceVPX fully defines the SpaceUM operation, signals, and connector with profiles provided for various combinations of voltages that may be supplied to each module. These may also be applied to power supply modules. Unlike OpenVPX with its maximum 560W module specification, SpaceVPX limits modules to no more than 100W.

SpaceVPX also defined the protocols for controlling the SM interface (SMI). It allows two options. One uses a subset of the VITA 46.11 IPMB protocol. This requires an intelligence to respond to each inquiry in under 3.3W which may be an issue for many simpler modules or for modules with large integrated processors. Thus SpaceVPX also defined a direct access protocol which uses direct access over the SMI to registers contained at each target. These registers provide basic information about the module and basic health readings like operational, temperature, voltage and built in test results. It is expandable so it has the hooks necessary to access other parts of a module that are so connected.

At the next level, SpaceVPX also defines the connections between slots in its backplane profiles. These are created to handle the maximum size for the given topology (such as data plane mesh or data plane star) and show how slots should be wired to one another. As before, these draw heavily from the OpenVPX heritage but are personalized to span the set of slots defined in SpaceVPX. If a user needs less slots (e.g. only has 6 payloads with an 8 slot switch), slots may be eliminated. Peripheral slots may be added to any part of a backplane since they do not contain any data plane connections. However, if they use control or utility plane signals, they must follow the rules for other slots in that profile and make sure the controller provides sufficient drops to service their needs.

The top level of standardization is at the chassis level. In OpenVPX, all chasses are defined for development usage. In SpaceVPX, chasses are also defined for flight usage. Primary voltages and the size of the slots are defined at this level. All profiles (slot, module, backplane and chassis) receive a label that accurately describes its makeup in a single label.

V. Interface Mappings

Figure 1 shows a full slot profile for a SpaceVPX controller slot. Other slot profiles in the Payload family of slot profiles are subsets of this. RapidIO (connector segment P1) is currently the only data plane and SpaceWire (connector segments P3 and P4) is the only control plane defined in SpaceVPX. The expansion plane may be used for either additional RapidIO lanes or any number of user defined I/O. Many backplane profiles define a slot to slot daisy chain using the P2 interface. Special capability is defined in P5 to provide a CompactPCI bus that may be daisy-chained to peripheral modules built of either SpaceVPX or CompactPCI form factor. P6 is used by the controller to route the SM interface, resets and common clocks to up to four modules or to the SpaceUM module to select between A and B controllers.

SpaceWire is fully defined as the control plane in SpaceVPX and ports are defined on this controller slot. If less SpaceWire ports are available, they should be depopulated starting with the top of P3 downwards. A minimum SpaceVPX implementation of the control plane for any logic module requires only two SpaceWire ports, routed from each controller in the system.

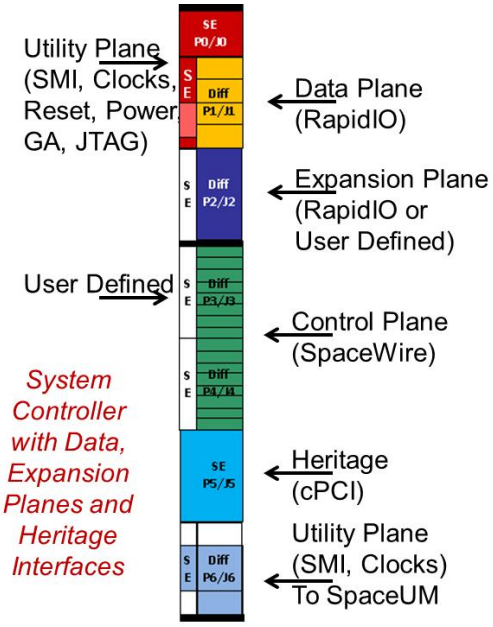


Figure 1: Interface Planes Mapped to Slot Profiles

SpaceWire as a control plane can be used for moving around configuration data and code, handling updates of module memory, collecting telemetry and status from modules beyond the 400 Kbps of the SMI in the utility plane. SpaceWire may also be used for medium speed data transfer. Since SpaceWire ports are capable of up to 400 MHz operation, this could be used for many data handling operations that don't need the full performance of SERDES circuitry and RapidIO ports. All defined backplane profiles show SpaceWire topologies as stars or radially-driven from each controller. However, with the minimum two ports on each module, a daisy chain architecture is possible. For fault tolerance, four ports would be an optimal minimum so that two separate daisy chains could be maintained in a system to allow working around errors.

The 2008 version of SpaceWire [20] is currently specified to run in SpaceVPX modules up to the data link layer using the SpaceVPX backplane and connector as the physical layer. Once the updated SpaceWire standard is released, analysis is needed to make sure this meets the Type B requirements. Higher level layers are currently left up to the user.

For many systems, SpaceWire will not provide enough bandwidth. RapidIO version 2.1 has been defined as the data plane with both switched and mesh topologies included to provide sufficient and scalable data moving bandwidth.

The RapidIO protocol is an international standard that is regularly updated by the RapidIO Trade Association. The protocol is designed as peer-to-peer, with a central controller used to configure and enumerate the network at the time of start-up. The RapidIO physical layer (PHY) is based on SERDES circuitry with encoding of data into characters to achieve balance over the long term. With revision 3.0 of the specification, the baud rate per lane was extended to 10.3125 Gbaud and for this baud rate and those above it the encoding mechanism was updated from the standard 8b/10b to 64b/67b, significantly decreasing the associated overhead. As of June 2016 with revision 4.0, the top baud rate has been extended to 25 Gbaud/lane. Valid port widths are 1, 2, 4, 8, and 16 lanes, although all commercial products to date support port width only up to 4 lanes. The protocol includes basic read, write, and maintenance functions, but also supports a number of optional features that address the needs of specific markets and users.

Updates to the specification are developed by task groups under the RapidIO Trade Association. In 2012, a new task group was created specifically to address unique requirements of spaceborne applications. Comprised of both corporate and government representatives, the group defined a series of enhancements that were published in revision 3.1 in 2014 [21].

The group defined "space device profiles" that included some of the optional features as required for use in space, including the error management extensions and multicasting. The new space features include the following:

- Structurally asymmetric links simplify the previously added dynamic asymmetric link capability, based on the assumption that sources such as sensors will always transmit far more data than it is necessary for them to receive. Return information will primarily consist of commands, responses, and error messages.
- Fault tolerant enhancements for port width degradation vs. the previous capability that limited which lanes could be used when a port degraded. The enhanced capability allowed for a 4-lane to 2-lane transition using either lanes 0 and 1 or lanes 2 and 3, and also allowed for any of the 4 lanes to be used as a single lane.
- Multicast event control symbol (MECS) based time synchronization and distribution again simplified an existing time distribution mechanism to provide accuracy almost as high with far less added hardware.
- A multiple entry error log was also defined that would allow for the capture of the exact sequence of errors as they occurred. This allows for significantly greater diagnostic capability than the single entry baseline error log register.
- Pseudo-random binary sequence (PRBS) circuitry supports in-flight testing of links to determine issues with a port and allow determination which lane of a port is the source of difficulty. During this testing the port is not active. Once testing has been completed, the port can be restarted configured as required.

All of these enhancements are capable of being used outside of the space market. For that reason, they were embedded directly into the specification as opposed to being identified uniquely for use only in space.

Hybrid systems with data movement using RapidIO and SpaceWire are easily constructed using SpaceVPX profiles.

VI. SpaceVPX Modules

Figure 2 shows a SpaceVPX system with several representative module types focused on using SpaceWire for Control and Data. This system controls 6 instruments attached to the SpaceVPX chassis. The controller uses its 16 port router to control and move data between all other logic modules. Shown in green are BAE Systems ASSPs that could provide the SpaceWire interface functions. A single string solution could be created using all the solid modules. Redundant modules are shown and dashed lines connect these to the other modules. Utility plane distribution and cross-strapping is also shown out of the bottom of each module routed through the SpaceUM module(s). If a single string is used, only one SpaceUM is needed. In a redundant configuration, two SpaceUM modules are required. Note twice as many SpaceWire links are provided to the Mass Memory since that often requires more bandwidth to store and retrieve data from all the potential data sources.

Figure 3 shows an upgraded system where RapidIO is used for the data plane and SpaceWire continues to function as the control plane. Here many of the SpaceWire components have migrated to RapidIO components that also support SpaceWire interconnects. The Mass Memory now relies on RapidIO and the data plane for its data stream inputs and outputs. Note the data plane switch is implemented in a seventh logic module.

Also, more instruments may be supported by the higher speed external I/O.

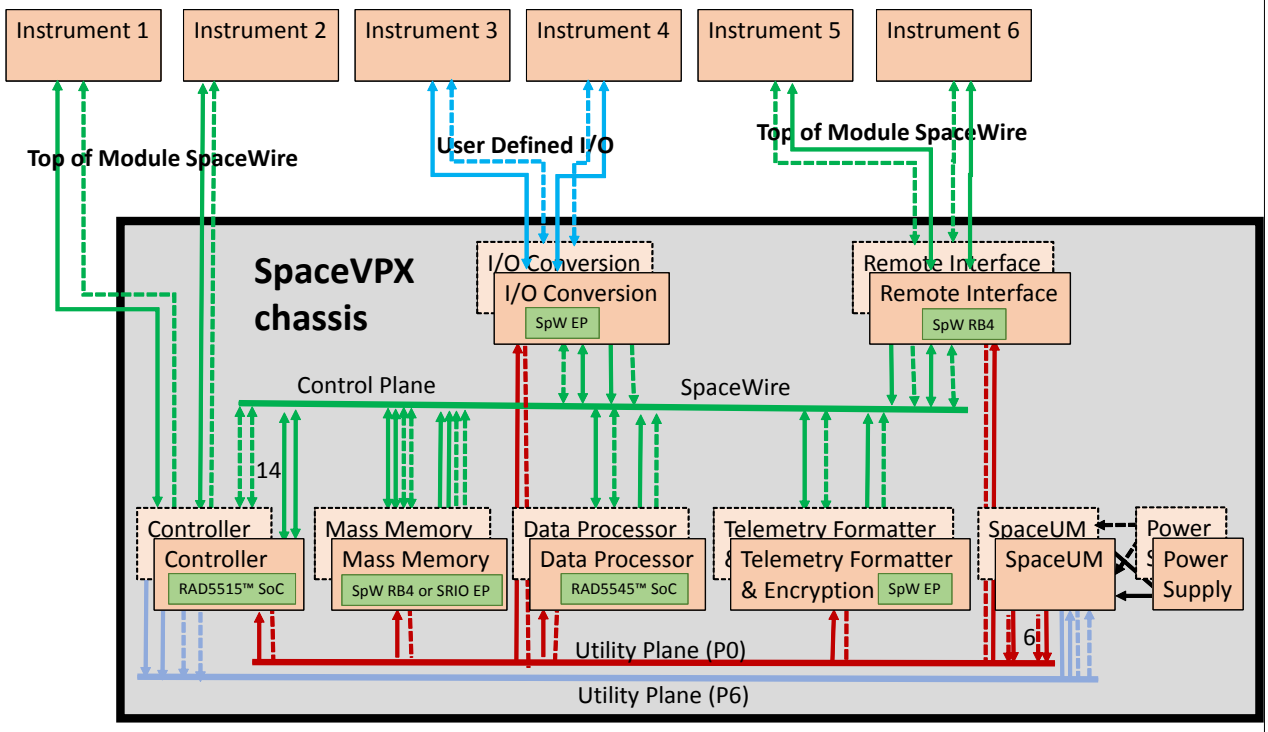


Figure 2: SpaceVPX system using SpaceWire for control and data

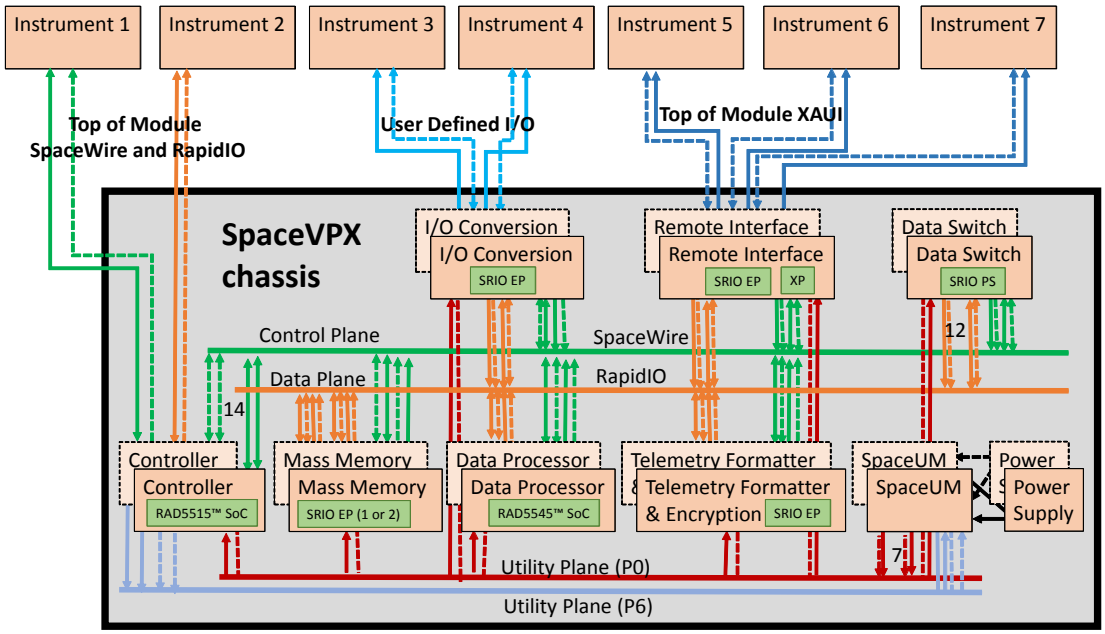


Figure 3: SpaceVPX system using SpaceWire for control and RapidIO for data

VII. Current Standards Efforts

A second standard effort, SpaceVPXlite, was started in 2015 and will become VITA 78.1 when approved. Its purpose is to focus on the 3U design space of SpaceVPX. 3U SpaceUM modules as defined in SpaceVPX can only control 2 logic modules versus the 8 logic modules controlled in a 6U system. A major emphasis of SpaceVPXlite has been to improve this overhead penalty and has led to the separation of the Utility plane signal switching from the Utility plane power switching functions by replacing SpaceUM modules with Power Switches and redefining the utility plane inputs to each logic module. Additional profiles are being added to match recent OpenVPX additions including optical and RF backplane connector options. The standard has mostly been written and is now in the working group review stages.

SpaceVPX is seeing widespread usage among spacecraft module developers. The NGSIS VITA working group continues to hold telecons on a weekly basis and discuss any shortcomings or corrections. As a result, a set of errata was published in May 2016 pointing out obvious errors. The group is preparing to start a minor revision to VITA 78 which will correct identified errors and omissions, pick-up some missing 3U content from 78.1 and add a few new elements to round out the standard. Also included will be an expanded user guide section to help first time users better navigate and use the standard.

VIII. Summary

SpaceVPX was developed to provide a standardized form factor for the next generation of high performance modules using interoperable SERDES-based fabrics with a focus on fault tolerance and scalability. It focuses on the use of RapidIO for high speed data movement and SpaceWire for command and data handling as well as medium speed data transfer. SpaceVPX doesn't forget its heritage and has elements that may interface to heritage system elements. Ratified in 2015, it is beginning to see widespread adoption that should lead to multiple interoperable modules that may be assembled into scalable high performance payloads and other spacecraft electronics modules.

References

[1] D. Rickard, et. al., "On-Board Networks with Radiation Hardened 45nm SOI Standard Components", Proceedings of the IEEE Aerospace 2015 Conference, Big Sky MT USA, March 2015.

[2] R. Berger, et. al., "Quad-Core Radiation-Hardened System-on-Chip Power Architecture Processor", Proceedings of the IEEE Aerospace 2015 Conference, Big Sky MT USA, March 2015.

[3] J. Marshall and R. Berger, "High Performance Network Components for Scalable Spaceborne Processing Needs", Proceedings of the 2016 International SpaceWire Conference, Yokohama, Japan, October 2016.

[4] J. Marshall, R. Berger and L. Assadzadeh "SpaceWire Fabric Used to Control Family of Standardized High Performance SpaceVPX Modules", Proceedings of the 2016 International SpaceWire Conference, Yokohama, Japan, October 2016.

[5] Berger, R. W., et. al., "RAD750 SpaceWire-Enabled Flight Computer for Lunar Reconnaissance Orbiter", Proceedings of 1st International SpaceWire Conference, Dundee, Scotland, September, 2007.

[6] R. Alena, P. Collier, M. Ahkter, B. Wood, S. Sinharoy and D. Shankar, "High Performance SpaceVPX Payload Computing Architecture Study", IEEE Aerospace 2016, Big Sky MT, March 2016.

[7] R. Merl and P. Graham, "MicroTCA for Space Applications", Proceedings of the IEEE Aerospace 2016 Conference, Big Sky MT USA, March 2016.

[8] "In Space", www.cubesatkit.com/content/space.html.

[9] Compact-PCI Base Specification, PICMG 2.0 Rev 3.0, October 1, 1999. www.picmg.org

[10] "CompactPCI Serial", PICMG CPCI-S Rev 2.0, June 12, 2015, www.picmg.org.

[11] "OpenVPX System Specification", ANSI/VITA 65-2010 (R2012), 2012, www.vita.org

[12] "American National Standard for VPX Baseline Standard", ANSI/VITA 46.0-2007 (R2013), November 2013, www.vita.org.

[13] "VPX REDI: Mechanical Specifications for Microcomputers Using Conduction Cooling Applied to VPX", ANSI/VITA 48.2, 2010, www.vita.org.

[14] "SpaceVPX Standard", ANSI/VITA 78.00-2015, www.vita.com

[15] "MicroTCA", PICMG MTCA.0 Rev 1.0, July 7, 2006, www.picmg.org

[16] "PC/104 Specification", Version 2.6, October 13, 2008, www.pc104.org.

[17] Collier, Charles Patrick, et al., "Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space", Proceedings of 2015 IEEE Aerospace Conference, Big Sky MT USA, March 2015

[18] J. Marshall, "Standardized SpaceWire Solutions for Next Generation Systems", Proceedings of the 2014 International SpaceWire Conference, Athens, Greece, September 2014.

[19] P. Collier, J. Marshall, R. Berger, M. Enoch, S. Goedeke, "Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space", AIAA 8 Reinventing Space 2013 Conference Proceedings, Los Angeles CA USA, September 2013.

[20] ECSS Standard ECSS-E-ST-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, July 2008.

[21] RapidIO Interconnect Specification 3.1, www.rapidio.org, September 2014

All figures are Copyright BAE Systems – used with permission

SpaceFibre 1 (Long)

Ruggedized Photonic Transceivers for Spacecraft Datalinks

SpaceFibre Session, Long Paper

Ronald T. Logan Jr.
Rugged Electronics and Photonics Division
Glenair Inc.
Glendale, California, USA
rlogan@glenair.com

Abstract — **Commercial-off-the-shelf photonic components designed for datacenter or industrial applications do not typically satisfy the environmental ruggedness requirements of aerospace applications. In order to reduce costs and schedule risk for insertion of photonic components into these harsh-environment applications, we developed ruggedized photonic transceiver modules for aerospace fiber-optic datalink applications up to 5 Gbps. We then performed reliability and environmental testing to demonstrate that these modules meet or exceed many of the requirements of these applications. In this paper we present performance characteristics and results of reliability and environmental tests for these transceiver components.**

Index Terms **— Relevant indexing terms: SpaceWire, SpaceFibre, Spacecraft Networking, Spacecraft Electronics, Spacecraft Photonics.**

I. Introduction

Data transmission requirements between avionics modules onboard spacecraft continue to increase, driven by the use of processors with high-speed serial data I/O to support the growing data requirements of advanced sensor systems and increased bandwidth of communications switches and satellite communications terminals. Optical fiber is an ideal medium for high-speed signal transmission on space platforms, since optical fiber cables support data rates up to many tens of gigabits per second (Gbps), are much lighter and smaller than copper wiring of equivalent bandwidth, are immune to radio-frequency (RF) interference from adjacent cables, and therefore require no RF shielding. The emerging SpaceFibre standard for spacecraft networking anticipates the use of high-speed fiber optic transmission between avionics modules and subsystems on spacecraft.

However, the availability of suitable photonic transceiver components for space applications is not widespread. The major manufacturers in the photonics industry are typically not able or willing to address the highly-specialized requirements, long design cycles, extreme environmental robustness, ultra-high reliability, traceability, radiation tolerance and small, inconsistent production volumes encountered with space applications. Conversely, the development of suitable transceiver hardware is typically beyond the engineering or budget capacity of most spacecraft programs. We believe this combination of factors has limited the adoption of photonic links on spacecraft, while multi-gigabit links have proliferated in non-space aerospace applications. We therefore undertook development of photonic transceivers designed to address the emerging aerospace requirements.

In this paper we will briefly review the components of photonic transmitters, receivers and transceivers, and highlight the challenges with spacecraft transceiver design. We then describe the approach to design of rugged photonic transceiver developments and the results of performance and environmental tests appropriate for space avionics applications.

II. Background and Challenges with Spacecraft Photonic Transceiver Design

We first briefly review the design of photonic transceivers, which have two main sub-components: laser transmitter and photodiode receiver. The function of the transmitter is to convert electrical serial data bits to optical pulses, and the photodiode receiver converts optical pulses to electrical serial data bits. These functions are realized in multi-gigabit systems using opto-electronic semiconductor devices (laser diodes and photodiodes) and electronic integrated circuit (IC) amplifier and control-loop devices.

The transmitter employs a laser diode which is current-modulated to impress the electrical serial data onto an optical signal as a series of on and off states. Laser diode threshold current and modulation efficiency are strong functions of temperature. Many transmitters incorporate a power monitor photodiode to sample and measure the laser output power and maintain the average output power at a constant level using a feedback loop with the average laser current as a control point.

The electronic driver IC amplifies the electrical bit stream from standard logic-levels such as Common-Mode Logic (CML) typically used as I/O to and from microprocessors, field-programmable gate-arrays (FPGAs), etc., to the level required to modulate the laser current to achieve optical modulation at the optimum level. Since the optical modulation vs bias-current slope efficiency is also a function of temperature, a second control system is used to maintain proper optical modulation over the operating temperature range. Careful matching, calibration and tuning of the bias control and modulation control circuits are required to insure that high-speed transmitters at multi-gigabit rates operate within industry-standard specifications over temperature. There are variations on these approaches, but what is always true is that some form of control of the laser current and modulation depth is required if the laser temperature will vary in operation.

The receiver contains a PIN photodiode, transimpedance amplifier IC, and limiting amplifier IC. The transimpedance amplifier often contains an AGC circuit to maintain the output level in an acceptable range when higher-level optical input signals are present. The limiting amplifier may also contain a bandwidth-limiting element to improve noise performance at lower bit rates.

For bit rates up to 5 Gbps, the above laser diode, photodiode and IC components are available that operate from -40C to +85C without external thermal controls. Manufacturers of commercially-available lasers, photodiodes and transceiver electronic ICs do not typically have test data for the performance of their devices in radiation environments. This is a central challenge to realizing photonic transceivers for space applications.

III. Technical Approach

Most modern datacom transceivers and IC chipsets contain CMOS circuitry and memory to support bias control lookup, serial I/O monitor and control ports, etc. to conform to datacom networking interface standards. However, these transceiver products are typically board-mountable units that accept commercial-grade optical connectors, and do not need to operate in the harsh aerospace environment, including radiation. As such they are not typically suitable for use in space. In order to be suitable for aerospace applications, appropriate aerospace-grade connectors need to be accommodated and the semiconductors must withstand the radiation exposure levels.

One example of an optical transceiver form factor that satisfies many of these requirements is shown in Figure 1, called a Size #8 opto-electronic contact. These devices provide electro-optic conversion of high-speed data signals from electrical to optical format, or optical to electrical format, inside of a fiber-optic connector on an avionics module in standard size #8 connector cavities. Because of the very small package size (~20 x 6.5 mm), we developed opto-electronic circuits using very simple IC chip sets that provide only basic monitor and control I/O signals such as “transmitter enable”, “transmitter fault” and “receiver loss of signal (LOS).” The focus of the development was on fitting into the allotted form-factor, strictly complying with ARINC 801 optical contact float requirements, and surviving harsh aerospace environments.

These transmitter and receiver contacts may be inserted into ARINC 400 or 600 avionic-bay connectors, or into special front-insert D38999 or D-sub connectors (see Figure 2), to provide data translation between electrical and optical domains inside of a panel-mount connector on an avionics module. The optical fiber interface of the ARINC 801 fiber optic contact used supports repeated blind-mating due to the incorporation of a floating optical ferrule, by using a unique design that incorporates a flexible circuit board assembly internal to the unit [1].

The transmitter contact utilizes a hermetically-sealed GaAs VCSEL and the receiver a hermetically-sealed GaAs photodiode at 850nm with a multi-mode ARINC 801 fiber optic interface.

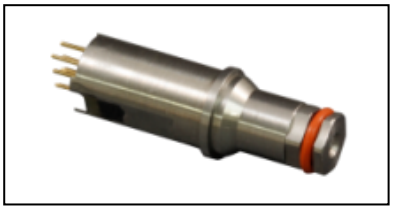


Figure 1. Opto-Electronic Contact.

Figure 2. Size #8 contacts in panel-mount avionics connectors: D-sub (upper) and D38999 (lower).

The optical interface to the cable is accomplished using a mating adapter insert in the plug Size 8 cavity that accepts a standard ARINC 801 optical contact. These opto-electronic contacts can support data rates from 50 Mbps to 5 Gbps, and interface with standard Common-Mode-Logic (CML) differential data signal levels on the electrical inputs and

outputs. They operate from 3.3 V input power, consume ~60 mA of current, and have a transmitter enable input, as well as transmitter fault and Loss of Signal (LOS) output status discrete signals. The optical interface specifications conform to the output power levels, eye-mask-margins, extinction ratios, and receiver sensitivity typical of industry-standard Fiber Channel and Gigabit Ethernet specifications, so the optical ports will interface via standard 50/125 micron or 62.5/125 micron multimode optical fiber with other commercial datacom optical transceivers as might be encountered in ground test equipment.

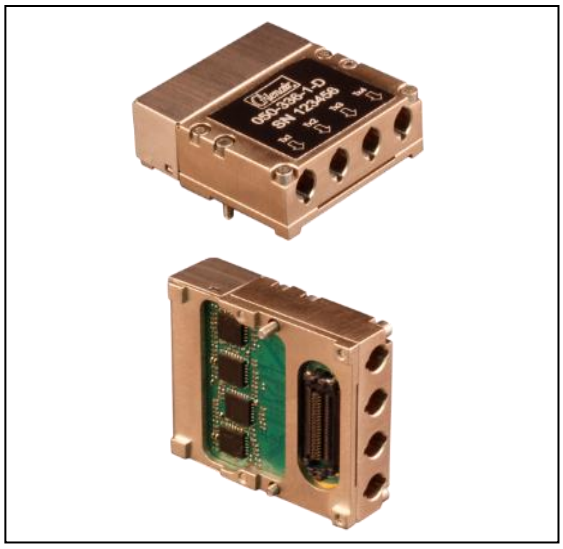


Figure 3. PCB-mountable quad-output transmitter unit: Top view (upper) and bottom view (lower).

In addition to the Size #8 contacts, the same optical and electrical device circuits have been incorporated into printed-circuit-board (PCB) mountable transceivers as shown in Figures 3 and 4. These devices utilize a high-speed surface-mounted PCB connector on the bottom of the unit to provide the connectivity to the host PCB via 100-ohm differential CML data streams, and are affixed using captive screws to threaded inserts that are soldered into the host PCB. The four optical interfaces of the four-fiber version in Figure 3 are machined cavities that strictly conform to the ARINC 801 standard, with retaining clips to hold the contact that require the use of an extraction tool for contact removal.

The two-fiber form-factor shown in Figure 4 utilizes a new connector developed by Glenair (Glenair GC-type) that has extremely low mass, low protrusion and very high tolerance to shock and vibration. This connector and transceiver permit a small footprint to be consumed on the customer PCB, and are much smaller than a standard datacom SFP pluggable transceiver, as shown in Figure 5.

One benefit of a simplified circuit approach is that there are no microprocessor or memory devices in the units, which are typically more susceptible to single-event effects (SEE), latch-up, etc.

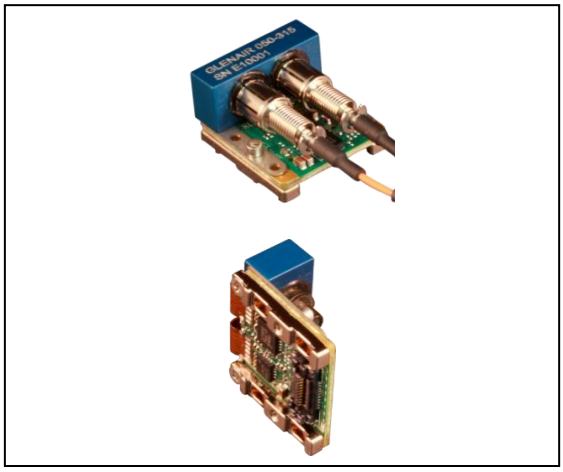


Figure 4. Two-fiber PCB-mount transceiver form-factor. Top view (upper) and bottom view (lower.)

Figure 5. Size comparison of Glenair 2-fiber transceiver with commercial datacom SFP pluggable transceiver.

IV. Test Results

Various reliability and qualification tests were conducted on the parts described above. Some key results are summarized here.

The filtered transmitter eye diagram at 4.25 Gbps for the Size #8 contacts at various temperatures is shown in Figure 6, showing stable optical power, acceptable eye-mask margins and extinction ratios over the -40C to +90C range of ambient operating temperature. The performance of the other transceiver form-factors is similar, since they use the same circuit schematic and components. The eye-mask testing was performed at 4.25 Gbps due to the availability of test equipment with this data rate filter. The links tested using these devices also run error-free at 5 Gbps.

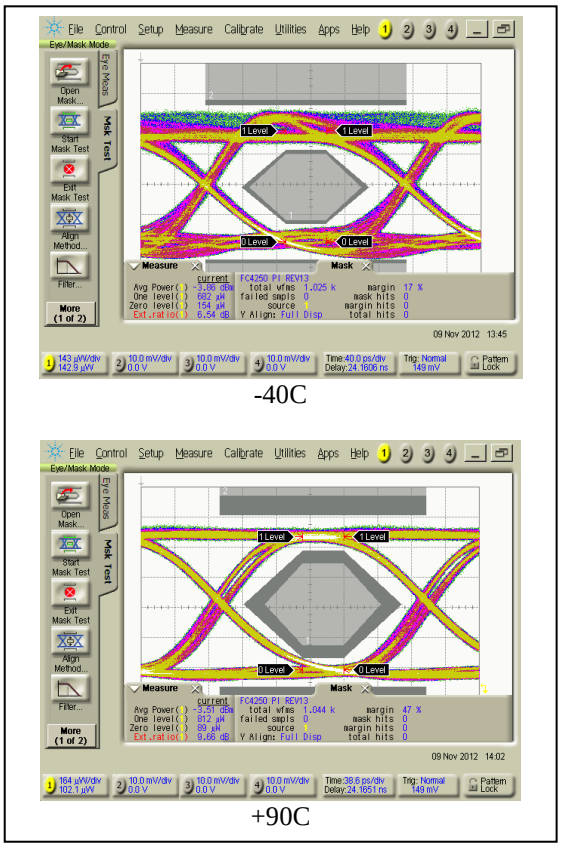


Figure 6. Size #8 contact filtered eye diagrams at 4.25Gbps.

The receiver sensitivity typical for the Size #8 opto-electronic contact measured at 4.25 Gbps at various temperatures is shown in Figure 7. As evident in the figure, there is approximately -19 dBm, which is 5 dB of margin beyond the Fiber Channel standard specification for 4.25 Gbps of -14 dBm. Given the transmitter output power of approximately -3.5 dBm, this yields an optical link budget of greater than 16 dB at 4.25 Gbps.

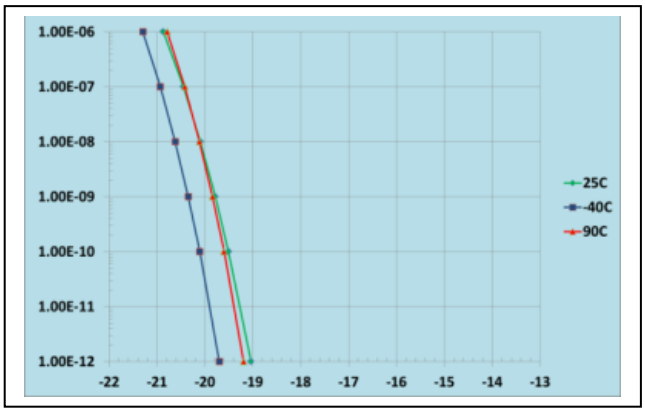


Figure 7. Receiver sensitivity at 4.25Gbps.

Accelerated aging tests were performed on 20 transmitter and receiver devices while operating at +85C, and the results are shown in Figure 8. No failures were observed.

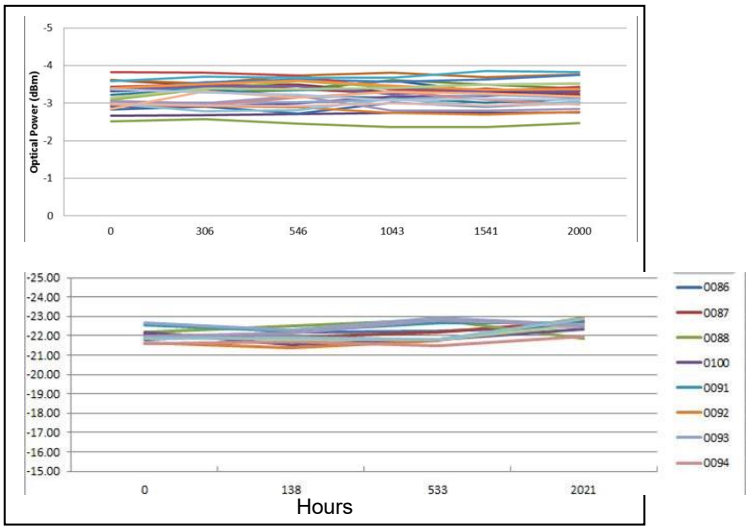


Figure 8. Accelerated aging of Size #8 opto-electronic contacts. Transmitter output power (top) and receiver sensitivity at 1.25 Gbps (bottom).

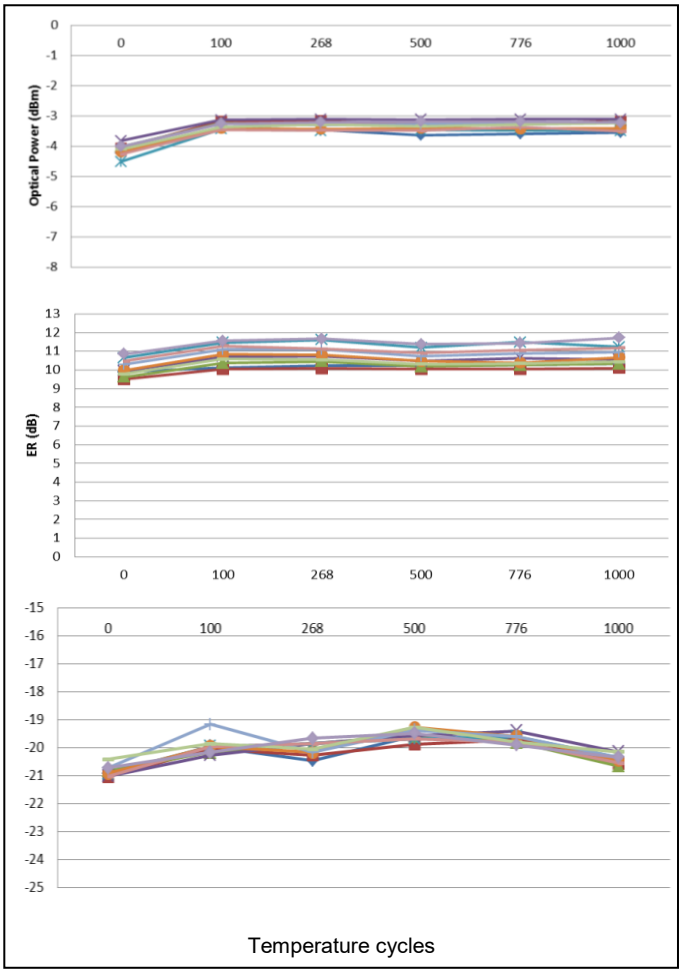


Figure 9. Thermal cycling test from -55C to +125C for Glenair PCB-mount transmitter output power and extinction ratio and receiver sensitivity at 4.25 Gbps. The units were removed from the test chamber at the intervals indicated.

Temperature cycling testing was performed for 1000 cycles from -55C to +125C, non-operating on the PCB-mount transceivers and the Size #8 contacts. The units were removed at intervals and subjected to full production test regimen over temperature from -40C to +85C to insure that the units were still within specifications.

Both styles of PCB-mounted transceivers, (ARINC 801 4-fiber and GC 2-fiber types) were subjected to operational vibration testing to a level of 54 Grms, with spectrum as indicated in Figure 10. The duration was 2 hours per axis, with data running and errors being monitored at 5 Gbps. No errors were detected.

This was followed by 650 G, 0.9 ms shock pulses, 10 shocks per direction in all three axes. The units were exposed to these levels while operating and errors were monitored at 5 Gbps. No errors were detected during any of these exposures.

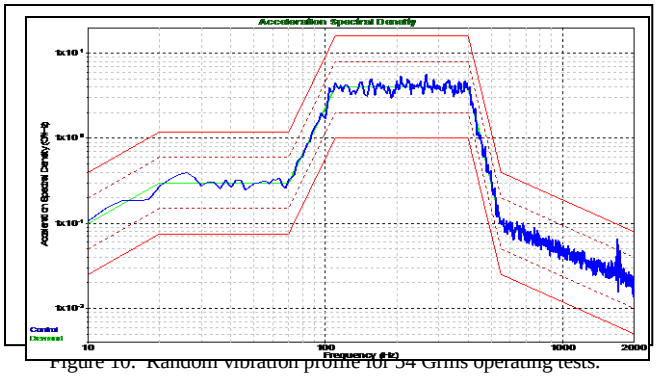


Figure 10. Random vibration profile for 54 Grms operating tests.

Finally, the Size #8 contacts were tested for resistance to radiation exposure to 165 krad of gamma radiation from a cobalt-60 source, and 2.5 x 10^{12} neutrons/cm^2, while operating under continuous error monitoring, with no errors detected.

Future test plans include charged-particle testing with protons and heavy ions, and will be reported in future publications.

I. Conclusions

Compact, rugged, opto-electronic transmitters, receivers, and transceiver modules in various form-factors were developed and tested to 5 Gbps data rates during various harsh environmental exposures. These transceivers were designed to interface with aerospace-grade fiber-optic connectors suitable for space-flight applications. These devices were subjected to various tests, including thermal cycling, high vibration and shock, and gamma and neutron radiation, and found to survive with no data errors. Further testing is planned.

Acknowledgment

Glenair thanks Technical University Munich for the gamma and neutron radiation testing performed on Glenair products.

References

[1] US Patent 9,297,972 Ronald T. Logan Jr., Sean Zargari, Mehrdad Ghara, Huan Do, “Advanced fiber-optic contact and method,” issued 3/29/2016.

SpaceFibre Networks

SpaceFibre, Long Paper

Steve Parkes, Chris McClements, David McLaren,
Space Technology Centre, University of Dundee,
166 Nethergate, Dundee, DD1 4EE, UK
smparkes@dundee.ac.uk

Albert Ferrer Florit, Alberto Gonzalez Villafranca,
STAR-Dundee Ltd.,
STAR House, 166 Nethergate, Dundee, DD1 4EE, UK

***Abstract*— SpaceFibre [1][2][3] is the next generation of SpaceWire [4] on-board data-handling network technology for spaceflight operations, which runs over both electrical and fibre optic media. SpaceFibre has many benefits compared to SpaceWire, including much higher data-rates, integrated quality of service, fault recovery capabilities, multi-laning with graceful degradation and hot and cold redundancy, and low-latency broadcast messages that can carry 8-bytes of user information. Importantly SpaceFibre is backwards compatible with SpaceWire at the network level, allowing existing SpaceWire equipment to be incorporated into a SpaceFibre network without modification. SpaceFibre networks have been defined by the University of Dundee and STAR-Dundee, and incorporated in the network layer definition of the current draft SpaceFibre standard. STAR-Dundee has designed a SpaceFibre routing switch to evaluate various routing concepts, validate the standard specification and demonstrate a complete SpaceFibre network. A demonstration system has been built and key parts of the SpaceFibre network technology have been demonstrated.**

***Index Terms* — SpaceFibre, SpaceWire, Networking, Spacecraft Electronics.**

I. Introduction

SpaceFibre is the next generation of SpaceWire technology for spacecraft on-board data-handling. It is able to operate at multi-Gbits/s over distances of up to 5 m using electrical cable and 100 m using fibre optic cable. It is galvanically isolated, includes quality of service and fault detection, isolation and recovery capabilities. SpaceFibre is backwards compatible with SpaceWire at the Network level, which enables existing SpaceWire equipment to be connected into a SpaceFibre network without modification. Furthermore SpaceFibre has been designed to have a small footprint, enabling its implementation in flight qualified FPGAs and ASIC devices without using a large part of the device.

This paper outlines the operation of SpaceFibre networks, describes the SUNRISE SpaceFibre routing switch, and summarises the results of tests with this routing switch.

II. SpaceFibre Links

A. Links and Lanes

A SpaceFibre link is made up of one or more lanes, which carry information from one end of the link to the other. SpaceFibre lanes can run over an electrical or fibre optic physical layer. In a multi-lane link, some of the lanes can be unidirectional provided that at least one lane is bi-directional [5]. The SpaceFibre link provides quality of service and error recovery [3].

B. SpaceFibre Virtual Channels

SpaceFibre links carry traffic (application information) through one or more virtual channels. There is a maximum of 32 virtual channels on a link, which are numbered consecutively starting at 0. Traffic entering virtual channel N comes out of virtual channel N at the other end of the link.

Each virtual channel is provided with a quality of service (QoS) which has three components: bandwidth reservation, priority and scheduling. Bandwidth reservation, reserves a portion of the link bandwidth for the virtual channel. Priority assigns a priority-level to the virtual channel so that higher priority virtual channels are able to send before lower priority ones. Scheduling divides time into 64 sequential time-slots and specifies in which of those time-slots a virtual channel is permitted to send information. These three different QoS components are not alternatives, they work together. [3]

III. SpaceFibre Networks

In this section the operation of a SpaceFibre network is described.

A. SpaceFibre Packets

SpaceFibre packets are identical to SpaceWire packets. They are formed from data characters, end of packet markers, and error end of packet markers, as illustrated in Figure 1.

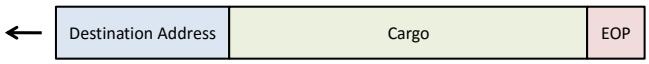


Figure 1 SpaceWire Packet Format

The "Destination Address" is the first part of the packet to be sent and is a list of data characters that represents either the identity of the destination node or the path that the packet has to take through a SpaceFibre network to reach the destination node. In the case of a point-to-point link directly between two nodes (no routers in between) the destination address is not necessary.

The "Cargo" is the data to be transferred from source to destination. Any number of data bytes can be transferred in the cargo of a SpaceFibre packet.

The "End_of_Packet" (EOP) is used to indicate the end of a packet. The data character following an End_of_Packet is the start of the next packet. There is no limit on the size of a SpaceFibre packet. “Error End of Packet” (EEP) is a form of

EOP which is used to indicate the premature end of a packet due to the occurrence an error.

B. SpaceFibre Virtual Networks

A SpaceFibre network is effectively a set of independent parallel SpaceWire networks. These parallel, independent networks are called "SpaceFibre virtual networks". Each virtual network runs over its own, distinct set of SpaceFibre virtual channels, comprising a virtual channel across each link used by the virtual network. Several virtual networks can then operate concurrently over a single physical SpaceFibre network. The overall physical network and the collection of virtual networks that run over that physical network is called the "SpaceFibre network".

The traffic running over each virtual network is constrained by the SpaceFibre quality of service mechanism to remain within its allocated bandwidth and to observer the priority and schedule allocated to it. A virtual network is able to opportunistically use more bandwidth than it has been allocated, when no other virtual network has traffic to send over the links of the SpaceFibre network that the particular virtual network wants to use.

As far as the addressing of packets and their routing across the network is concerned, SpaceFibre operates in the same way as SpaceWire. This has the substantial advantage that existing application software or SpaceWire equipment can be used with a SpaceFibre network by simply tying a SpaceWire link interface to a SpaceFibre virtual channel interface. The application does not need to know that it is running over SpaceFibre, but gains all the QoS and FDIR advantages of SpaceFibre. This make the integration of existing SpaceWire equipment both simple and advantageous.

C. Packet Addressing

SpaceFibre uses both path and logical addressing, which operate in the same way as SpaceWire. It is not possible to route a packet between two different virtual networks in a routing switch. As already stated virtual networks on a SpaceFibre network are like a set of parallel, independent SpaceWire networks. The packet routing is within one virtual network.

Path addressing uses the leading data character of a packet to determine how the packet should be routed at the next routing switch. If the value of the leading data character is in the range 0 to 31, it determines which port of the routing switch the packet will be forwarded through. For example, if the leading data character is 2, the packet will be forwarded through port 2 of the routing switch. If the leading data character is 0, it will be routed to port 0, the internal configuration port of the routing switch. If the leading data character is 31 and there are only 9 ports in the router, the packet will be discarded. Note that the ports of a router are number consecutively, starting at 0 for the internal configuration port.

If the leading data character is in the range 32-255, it is a logical address. The value of the leading data character is then used as the index into a routing table, which once configured, determines which port the packet is to be forwarded through. For example, if the leading data character is 40 and the entry in the routing table for index 40 contains the value 3, the packet will be routed to port 3 of the router. The routing table is configured using RMAP commands sent to the router configuration port [6]. Before configuration of the routing table has been done, any logical address will result in the packet being discarded. Path addressing operates at all times, before and after the routing table has been configured.

D. Fills

SpaceFibre runs much faster than SpaceWire, so requires an interface to the application which is wider than that of SpaceWire to carry the extra data. The interface to a SpaceFibre port is typically 32-bit wide or a multiple of 32-bits, whereas SpaceWire is 8-bits wide. If SpaceFibre is to send a packet which is not a multiple of 32-bits, the start of the packet or its tail end can be filled with Fill characters to make it 32-bit aligned. Therefore, a SpaceFibre data word contains four data characters, EOPs, EEPs or Fills. The use of Fills is illustrated in Figure 2 and Figure 3, where P represents a path-address data character, D represents a data character, E an EOP or EEP, and F a Fill.

Filling the start allows for a 32-bit aligned cargo, when path addressing is being used, as illustrated in Figure 2.

F	F	F	P
P	P	P	P
D	D	D	D
D	D	D	D
E	F	F	F

Figure 2 Fills at the start of a SpaceFibre packet

Fill characters are added at the beginning of a packet, to align a path address which is not a multiple of four data characters in length or to fill spaces that were previously occupied by a path address. This allows the leading SpaceFibre path address bytes to be removed by a router and replaced by Fill characters in order to keep the word-alignment of the SpaceWire cargo when it arrives at the destination. It also allows some fills to be added to the start of a packet to ensure that the cargo of the packet is 32-bit aligned when there is a path address that is not a multiple of four data characters.

Filling the end allows for the cargo to be any number of N-Chars, not a multiple of four N-Chars, as illustrated in Figure 3.

D	D	D	D
D	E	F	F
F	F	P	P
D	D	D	D
D	D	D	D
E	F	F	F

Figure 3 Fills at the end of a SpaceFibre packet

The Fill character is used in a data word containing an EOP or EEP to fill otherwise empty characters that follow the EOP or EEP. The above example shows two small packets in part of a frame being aligned to 32-bits.

E. Virtual Network Masters

A "network master" is a node on a SpaceFibre virtual network which is a source of SpaceFibre packets able to send packets autonomously, i.e. without first receiving a request

from another node. Note that a network master is different to a network manager, the latter is a network master that configures, controls and monitors the status of the entire SpaceFibre network.

If there is one network master on a virtual network then that virtual network can be deterministic. For example, the network master might be a control processor sending Remote Memory Access Protocol (RMAP) packets to other instrument nodes to control them and collect data from them, using RMAP. The traffic on the virtual network is controlled by the one network master node. The set of virtual channels that the specific virtual network runs over is allocated the bandwidth and priority according to its needs. If the virtual network is to provide time-bound determinism, its virtual channel will also be scheduled by the SpaceFibre QoS mechanism.

Within a single SpaceFibre virtual network, if there are two independent network masters, it is possible that they both send a packet to the same node, or through the same link to a router and then on to different nodes. Whenever these two network masters want to send a packet over the same link at the same time, there is a "collision" and one packet will have to wait for the other one to be sent. This is the same as the temporary "packet blocking" that can occur in a SpaceWire network. Each SpaceFibre virtual network operates just like a separate SpaceWire network, including temporary packet blocking.

Now, in some applications the temporary network blocking was a real pain in a SpaceWire network, especially if long packets were being used. Traffic from one application could delay traffic from another one, which could be difficult to handle under some circumstances. SpaceFibre solves this problem, by having multiple, independent virtual networks. If there is a single network master on each of these virtual networks, the packet blocking is avoided completely. It is still possible to have multiple network masters on the same virtual network, provided that packet blocking is not an issue for the traffic flowing over that virtual network, or provided that another mechanism is used to control the flow of traffic over that network.

This approach maintains full backwards compatibility with SpaceWire at the network level, which is essential if the large legacy of existing SpaceWire equipment is not to be squandered. Reuse of existing, proven equipment, reflected by the Technology Readiness Level (TRL), is an important way of improving reliability and reducing the cost of space missions. SpaceFibre offers a path for substantially upgrading the capabilities and performance of an onboard network without losing that valuable legacy.

IV. Reference Architecture

It is worth considering an example of how the virtual networks might be used in a typical space mission. First, a reference architecture is described.

A. Earth Observation Reference Architecture

A reference architecture has been devised which is representative of a typical high data-rate Earth Observation mission. This architecture is illustrated in Figure 4.

Instruments 1 and 2 are high data-rate instruments each with a SpaceFibre interface. They are connected via two SpaceFibre routers to the mass-memory unit which has two SpaceFibre interfaces. Each instrument is able to transfer data at up to 2 Gbits/s using a 2.5 Gbit/s SpaceFibre link.

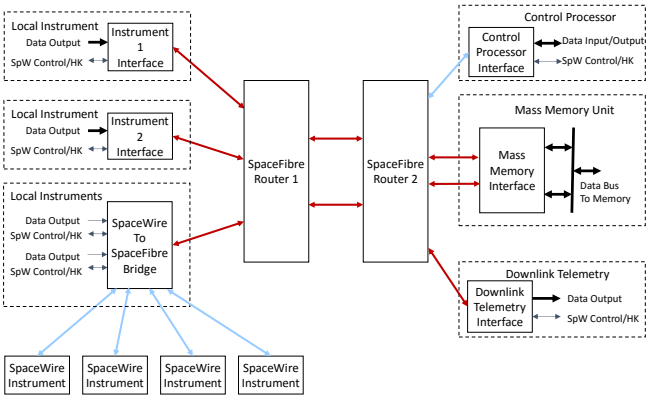


Figure 4 SpaceFibre Earth Observation Mission Reference Architecture

Four existing SpaceWire instruments are attached to a SpaceWire to SpaceFibre bridge device, each via a separate virtual channel of the SpaceFibre interface. Data from these SpaceWire devices is sent over the SpaceFibre network to the mass-memory unit.

Data from the mass-memory unit is passed to the downlink telemetry unit.

A control processor is able to access all of the instruments, the mass-memory unit and the downlink telemetry unit along with the SpaceFibre routing switches to configure and control the devices and to read housekeeping information from them.

The architecture in Figure 4 does not really need two routing switches, but two are included in the reference architecture to make it more generic.

B. Example Allocation of Virtual Networks

There are several functions that need to be carried out by the reference architecture of Figure 4. These functions are listed below:

1. SpaceFibre network management: configuring, monitoring and reconfiguring the SpaceFibre network;
2. Payload management; instrument control and status monitoring (housekeeping);
3. Data-handling system management; control and status monitoring (housekeeping) of the mass-memory unit and the downlink telemetry unit;
4. Sending data from the high data-rate Instrument 1 to the mass-memory unit
5. Sending data from the high data-rate Instrument 2 to the mass-memory unit;
6. Sending data from the four SpaceWire instruments to the mass-memory unit;
7. Sending data from the mass-memory unit to the downlink telemetry unit.

Each of these functions could be allocated a separate virtual network, requiring a total of seven virtual networks in the routing switches and mass-memory unit. Since there is only one control processor (ignoring a possible redundant unit), it is necessary to run the SpaceFibre network management, the payload management and the data-handling functions on the same processor. These functions can then share a virtual network since there will always only be the one control processor using that virtual network. This reduces the number

of virtual networks required to five. The five parallel virtual networks are illustrated in Figure 5.

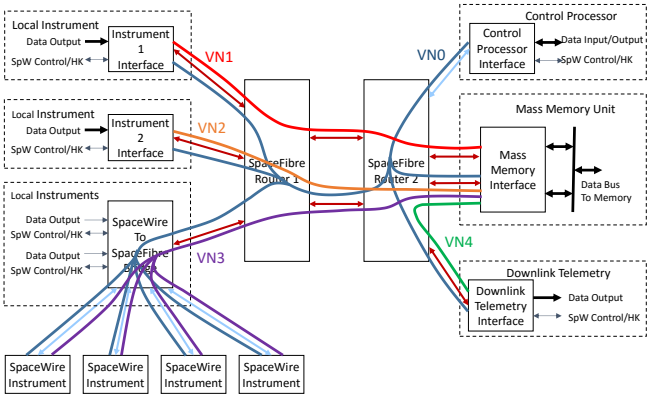


Figure 5 Parallel Virtual Networks

The control processor performing the network management, payload management and data-handling system management, uses one virtual network (VN0) and is able to access all of the instruments, routing switches, mass-memory and downlink telemetry units, over that one virtual network.

Instrument 1 uses another virtual network (VN1) to send data to the mass-memory unit. Similarly instrument 2 uses VN2.

The SpaceWire instruments all share one virtual network (VN3) for sending data to the mass-memory unit. This means that they will compete for access to the virtual network, as if they were running over a SpaceWire network.

C. Networks with Large Number of Nodes

When there are a large number of nodes in a network, it is possible to handle them in several different ways.

Firstly, a single virtual network could be used for several nodes which all act as network masters. It is simply accepted that within this virtual network temporary packet blocking will occur and will not be a problem for the applications related to those nodes. This virtual network operates the same as a SpaceWire network

Secondly, it is possible to increase the number of virtual channels so that there is one for each SpaceWire instrument. This depends on the number of virtual channels available in the SpaceFibre routers and mass-memory unit. In any case there is a limit to the maximum number of virtual networks that can be used. There is actually a maximum of 32 virtual channels over a link and 64 virtual networks across a SpaceFibre network.

A third alternative is to use one network master on a virtual network to handle all the communication for the nodes on that network. The configuration, control and housekeeping network is an example of this where there is one master node that uses RMAP commands to request information to all the nodes on the network including the configuration nodes within the routing switches.

A similar approach could be used for sending data from several instruments to the mass-memory unit. The mass interface controller could send out RMAP commands to request data from each of the SpaceWire instruments on a single virtual network in turn. For example the mass-memory unit could use VN3 to send RMAP commands to the SpaceWire instruments which respond with the requested data, which is then placed in memory.

Another possibility is to schedule the sending of information from the various equipment over a virtual network using time-slots, which are delimited by broadcast messages over the SpaceFibre network or time-codes on the SpaceWire network. Each equipment then sends its data in its allocated time-slot or time-slots.

D. Virtual Network to Virtual Channel Mapping

Virtual networks are mapped on to a set of virtual channels, one virtual channel for each link used by the virtual network. Each virtual channel on a link is mapped to one and only one virtual network. The virtual channel number used by a virtual network over one link does not need to be the same as the virtual channel number used on another link.

The simplest way of mapping a virtual network to a virtual channel is to use a one to one mapping, so that virtual network VN0 uses virtual channel VC0 on all of the links in the network. Similarly VN1 uses VC1 and so on. The problem with this simple approach is that it complicates the instrument nodes of the network. For example, a typical instrument will require two virtual networks; VN0 which is used for control and monitoring and another virtual network which is used for data transfer to a mass-memory unit. This is the case with instruments 1 and 2 in Figure 5, which use VN1 and VN2 respectively. If a mapping is done from the virtual network to the virtual channels, the hardware required in the instrument interfaces is simplified. For example, instrument 1 VN1 is mapped to VC1 and instrument 2 VN2 is mapped to VC1. This mapping needs to be done at both ends of the respective links. The routing switch then uses this mapping to route a packet to an output port on the same virtual network number as that on which the packet arrived. The virtual channel numbers may be different on the link over which the packet arrived and the link over which the packet is being forwarded, but the virtual network numbers mapped to these virtual channels are the same.

Using the example of Figure 5, the links running from Router 1 to Router 2 will carry instrument data from instrument 1 over VN1 and from instrument 2 over VN2. This data can go over either of the links between the two routing switches depending on the packet address. So over these links the following mapping applies:

- VN0 -> VC0, this is always the case
- VN1 -> VC1
- VN2 -> VC2
- VN3 -> VC3

VN4 does not use the links between the routers.

So VN2 is mapped to VC1 for the link from instrument 2 to router 1, because there are only two virtual channels available in the instrument interface. VN2 is then mapped to VC2 over the links from router 1 to router 2.

The virtual network to virtual channel mapping makes the routing switches more complex, because it has to handle the mapping, but makes the instrument interfaces simpler, because they normally only need two virtual networks, which can be supported by two virtual channels. The virtual network mapping also permits more virtual networks on a SpaceFibre networks than there are virtual channels on a SpaceFibre link, i.e. there are up to 64 virtual networks allowed in a network but only 32 virtual channels over a link. This is possible because

some virtual networks may use completely separate parts of the network.

V. SUNRISE SpaceFibre Routing Switch

A SpaceFibre router has been designed and implemented in the SUNRISE project funded by the UK Space Agency and STAR-Dundee. The architecture of this router is shown in Figure 6.

The SUNRISE router has eight SpaceFibre ports, numbered 1 to 8, each with four virtual channels. There is a configuration port (port 0) which is used for device configuration and which can be accessed using virtual channel 0 of any of the other ports. Another port (port 9) provides an interface to four SpaceWire ports using four virtual channels, one for each SpaceWire port. SpaceWire and SpaceFibre packets are switched by the routing switch in the same way, using the leading data character of a packet to determine the output port that the packet is to be switched to. Both path and logical addressing can be used with the SUNRISE router.

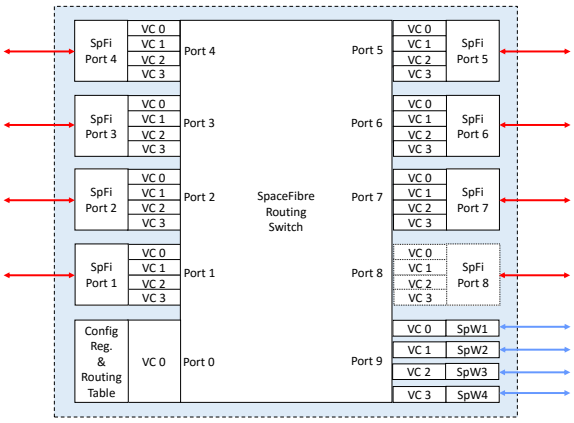


Figure 6 SUNRISE SpaceFibre Router Architecture

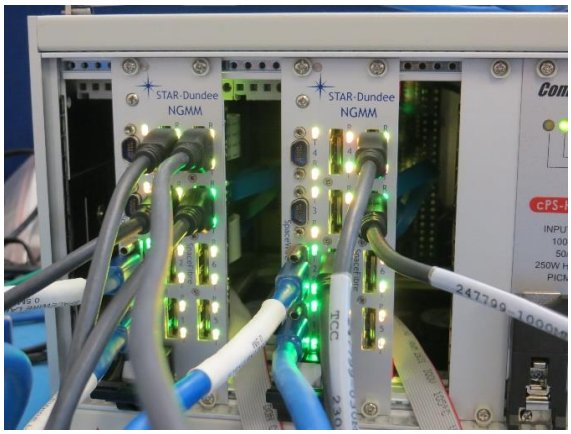


Figure 7 SUNRISE SpaceFibre Routers Under Test

The SUNRISE router was implemented initially in a Xilinx Spartan 6 FPGA. Two of the SUNRISE routers are shown under test in Figure 7. The SUNRISE routers are implemented on 3U cPCI/PXI boards. Power is taken from the backplane and the eight SpaceFibre and four SpaceWire ports are available on the 40mm wide front panel.

The SUNRISE router is now being implemented in a Microsemi RTG4 FPGA as shown in Figure 8 [7][8].

Figure 8 Prototype Microsemi RTG4 board for SUNRISE SpaceFibre Router

VI. Demonstration of SpaceFibre Network

The reference architecture has been implemented using a combination of radiation tolerant FPGAs and commercial FPGAs. This is illustrated in Figure 9.

Figure 9 SpaceFibre Network Demonstration

The equipment used in the demonstration system is detailed in Figure 10 [9].

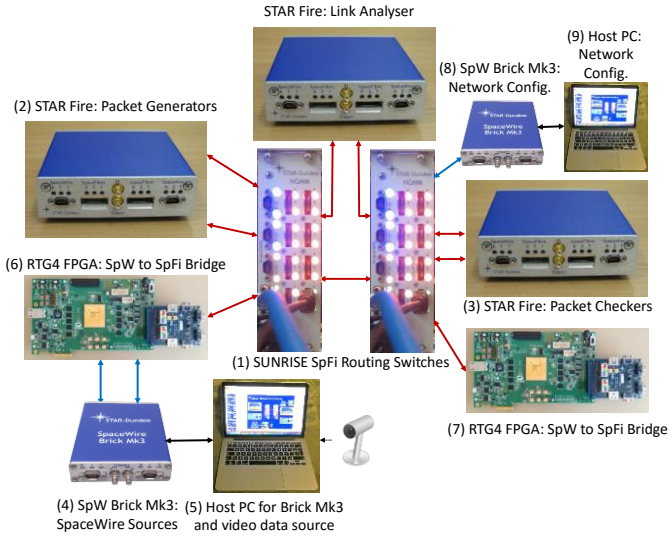


Figure 10 SpaceFibre Network Demonstration System

The following functions were demonstrated and validated:

- **SpaceFibre network operation:** using the two SUNRISE routing switches (1). Packets were successfully routed across the routing switches, remaining in their virtual networks.
- **High data-rate:** The STAR Fire unit has two SpaceFibre interfaces and incorporates packet generators that are able to generate SpaceFibre packets at the full data rate (2.5 Gbits/s) over each SpaceFibre link. STAR Fire unit (2) was used to simulate the two high data-rate instruments of the reference architecture. Data from these two "instruments" was sent across the network to the STAR Fire unit (3), which accepts and checks the high-data rate packets. STAR Fire unit (3) is acting like a mass memory accepting data from the high data-rate instruments.
- **SpaceWire to SpaceFibre bridging:** A SpaceWire Brick Mk3 (4) was used to generate two streams of SpaceWire packets under control of the host PC (5). The SpaceWire links are attached to a Microsemi RTG4 development board via a STAR-Dundee FMC board [7]. The RTG4 is programmed with a SpaceWire to SpaceFibre bridge design connecting four SpaceWire interfaces to four virtual channels of a SpaceFibre interface. One SpaceWire link is sending video data from a webcam attached to the host PC (5). The other SpaceWire link is sending packets from a SpaceWire packet generator running on host PC (5) to another PC (9) so that they can be checked for errors. The SpaceWire packets are converted to SpaceFibre packets, which is trivial as they have the same format, and sent across the SpaceFibre network. The video data is sent to another RTG4 board (7). The other data is sent via port 9 of a SpaceFibre router (1) which is a port where the virtual channels are connected to SpaceWire interfaces. This SpaceWire data goes across a SpaceWire link to another Brick Mk3 (8) and on to a host PC (9) where it is checked.
- **Quality of Service:** The STAR Fire packet generators (2) provide a total data rate of 2 x 2.5 Gbits/s, using all the network bandwidth between the two routing switches (1). The virtual channels they are using are assigned relatively low priority. The SpaceWire to SpaceFibre bridge (6) uses a virtual channel with higher priority. Whenever it wants to send data, it is able to do so, within the constraints of its allocated bandwidth. This is demonstrated by the real-time video data stream being transferred across the network.
- **Fault detection, isolation and recovery:** the link being used to transfer the traffic from the SpaceWire to SpaceFibre Bridge (6) between the two routing switches (1) can be unplugged. The video traffic then stops and the SpaceWire packet generated data stops. When the link is plugged back in the SpaceWire packet generated data continues and there is no loss of packets. The packets are checked for errors including missing packets in the host computer (9). While the link was disconnected no packets could be transferred but the packet being transferred when the link was disconnected was not lost. Clearly with the video data stream, data is lost once the buffers in the system are filled. The key point is that packets in transit across the network are not lost.
- **Network configuration:** the network is configured using the host computer (9) via a SpaceWire connection to the right hand routing switch (1).

For debugging and analysis purposes, a third STAR Fire unit (10) operating as a link analyser is included on one of the links between the two routing switches (1) [10].

VII. Conclusions

SpaceFibre networks have been defined by the University of Dundee and STAR-Dundee, and incorporated in the network layer definition of the current draft SpaceFibre standard. STAR-Dundee has designed the SUNRISE SpaceFibre routing switch to evaluate various routing concepts, validate the standard specification and demonstrate a complete SpaceFibre network. A reference architecture for a SpaceFibre network targeted at Earth Observation applications has been defined. A demonstration system has been built reflecting this reference architecture and key parts of the SpaceFibre network technology have been demonstrated.

Acknowledgments

The research leading to these results has received funding from the European Space Agency under ESA contract numbers 4000102641 and 17938/03/NL/LvH, from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement numbers 263148 and 284389 and from the UK Space Agency and CEOI-ST under University of Leicester contract numbers: RP10G0348A02, RP10G0348B206 and RP10G0348A207.

References

[1] S. Parkes, A. Ferrer Florit and A. Gonzalez Villafranca, "SpaceFibre Standard", Draft H5, University of Dundee, July 2016.

[2] S. Parkes, C. McClements and M. Suess, "SpaceFibre", International SpaceWire Conference, St Petersburg, Russia, 2010, ISBN 978-0-9557196-2-2, pp 41-45.

[3] S. Parkes et al, "SpaceFibre: Multi-Gigabit/s Interconnect for Spacecraft On-board Data Handling", IEEE Aerospace Conference, Big Sky, Montana, 2015.

[4] ECSS Standard ECSS-E-ST-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, July 2008, available from http://www.ecss.nl.

[5] A. Ferrer Florit, A. Gonzalez Villafranca and S. Parkes, "SpaceFibre Multi-Lane", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

[6] ECSS Standard ECSS-E-ST-50-52C, "SpaceWire – Remote memory access protocol", Issue 1, European Cooperation for Space Data Standardization, 5 February 2010, available from http://www.ecss.nl.

[7] S. Parkes et al, "SpaceWire and SpaceFibre on the Microsemi RTG4 FPGA", IEEE Aerospace Conference, Big Sky, Montana, 2016.

[8] S. Parkes, A. Ferrer Florit, A. Gonzalez Villafranca, C. McClements, B. Yu, P. Scott, J. Logan, D. Dillon and D. McLaren and "SpaceFibre Flight Equipment", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

[9] A. Gonzalez Villafranca, S. Parkes, C. McClements, B. Yu, P. Scott and A. Ferrer Florit, "A New Generation of SpaceFibre Test and Development Equipment", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

[10] S. Mudie and S. Parkes, "SpaceFibre Link Analysis", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

SpaceFibre Link Analysis

SpaceFibre 1, Long Paper

Stephen Mudie
STAR-Dundee Ltd
Dundee, Scotland
stephen.mudie@star-dundee.com

Steve Parkes
School of Computing
University of Dundee
Dundee, Scotland, UK
sparkes@computing.dundee.ac.uk

For those responsible for the design and implementation of a SpaceFibre network it is essential to be able to capture and view the traffic on a SpaceFibre link in order to help validate the link is operating as expected and debug the link should any unexpected behaviour be observed. STAR-Dundee Ltd have developed hardware independent SpaceFibre Link Analyser software for this purpose. This paper describes how the software views, combined with the traffic capture capabilities of the STAR Fire unit, can be used to perform SpaceFibre link analysis.

***Index Terms*— SpaceFibre, Link Analysis, STAR Fire**

I. Introduction

SpaceFibre [1][2] is a multi-Gbits/s, on-board network technology for spaceflight applications that will soon become a formal European Cooperation for Space Standardization (ECSS) standard. At present the SpaceFibre enabled STAR Fire unit [3] from STAR-Dundee Ltd has allowed users to transmit and receive simple data patterns and perform some basic SpaceFibre link analysis for prototyping purposes. Over the past couple of years work has been undertaken to replace the software provided with the STAR Fire to leverage new advanced data generators and checkers and to greatly improve on the SpaceFibre link analysis capabilities. One result of this work has been the development of the hardware independent SpaceFibre Link Analyser software. This paper aims to describe how using this software, those responsible for the design and implementation of a SpaceFibre network might perform SpaceFibre link analysis. This is very important in order to help validate a SpaceFibre link operates as expected and to debug any unexpected behaviour.

In this paper the key requirements of the software are described along with the hardware currently supported, and a short overview of how the software is controlled is provided along with a description of the triggering capabilities. Each of the different SpaceFibre traffic views are briefly described and the key features of these summarised. Screenshots of different SpaceFibre traffic scenarios captured and displayed using the SpaceFibre Link Analyser software are then presented as simple examples of its use.

II. Aims

Initial discussions regarding the SpaceFibre Link Analyser software highlighted a number of requirements outlined below.

- SpaceFibre traffic shall be captured when an event of interest occurs, for example a specific word such as a receive error or symbol such as EDF (End of Data Frame). This ensures the user captures the traffic they are most interested in.
- Three initial views shall be developed that display SpaceFibre traffic at the SpaceFibre symbol/word, frame and packet levels. This allows the user to thoroughly inspect the SpaceFibre traffic with varying levels of detail.
- The software shall be designed to support multiple device types capable of capturing/recording SpaceFibre traffic. This ensures continuity for users familiar with one device type switching to another. This also minimises custom development work for future devices with common functionality.

III. Hardware

The SpaceFibre Link Analyser software has been designed to support multiple device types with capture/recording capabilities. However, currently the STAR Fire is the only device type supported. The STAR Fire can transparently capture SpaceFibre traffic on a single link in both directions. Traffic capture can be triggered on detection of a SpaceFibre word, sequence of four symbols or an error. In addition to its capture capabilities, the STAR Fire can transmit and receive SpaceFibre traffic using data generators and checkers (these have recently been updated and now support more advanced data patterns), can route SpaceWire traffic over the SpaceFibre interfaces and can also decode SpaceFibre signals for use with a logic analyser.

Fig. 1. STAR Fire Unit

The STAR Fire has two SpaceFibre interfaces, two SpaceWire interfaces and two external triggers on the front of the unit. On the rear there are a further two external triggers, a USB 2.0 interface (soon to be USB 3.0), two MICTOR connectors and a power connector.

The STAR Fire is a great target for the SpaceFibre Link Analyser software as it provides the trigger and capture capabilities required, and can also act as a node for testing purposes, transmitting and receiving SpaceFibre traffic.

IV. Operation

Using the SpaceFibre Link Analyser software is quite simple. First the user should set the capture properties, for example the post trigger memory size. Next the capture trigger is configured, for example trigger on an SDF (Start of Data Frame) word. Then start SpaceFibre traffic capture (this causes the device to continuously capture to a circular buffer) and wait for the trigger to occur. When the trigger is detected or is forced by the user, the device memory is filled. The captured SpaceFibre traffic is then displayed in the different views.

V. Triggering

Triggering is used to ensure SpaceFibre traffic of interest is captured. Before a trigger occurs SpaceFibre traffic is captured to a circular buffer continuously. When the trigger occurs, and the post trigger memory is filled, the contents of the capture buffer are accessed by the software and displayed.

The SpaceFibre Link Analyser software can trigger on nothing, a sequence of four SpaceFibre symbols or a SpaceFibre word. When trigger on nothing is selected, the trigger immediately occurs when the user chooses to stop capture. When trigger on word is selected, the user selects the word type and can optionally specify properties specific to that word. This is shown below for the SDF word. In this example the trigger will occur when an SDF word is detected on virtual channel one.

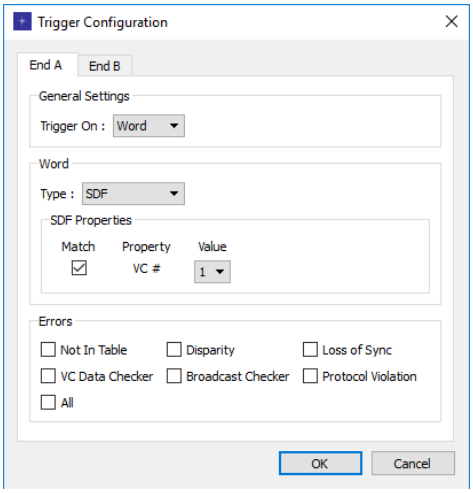


Fig. 2. Trigger on SpaceFibre Word

When trigger on symbol is selected, the user can select four consecutive symbols on which to trigger. This is shown in the screenshot below. In this example the trigger will occur when an SBF (Start of Broadcast Frame) word is detected on broadcast channel zero with sequence number five.

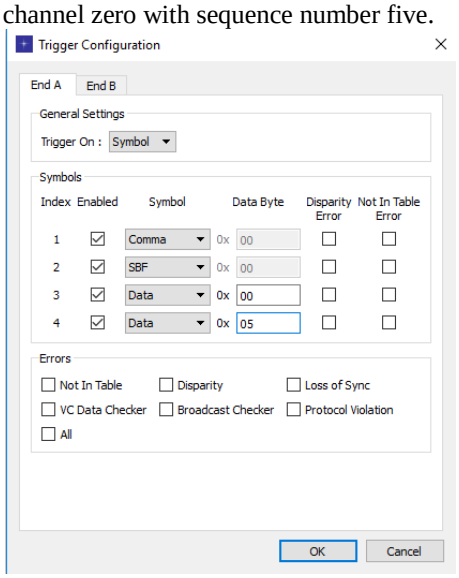


Fig. 3. Trigger on SpaceFibre Symbol

As is shown in the figures above, in addition to triggering on specific SpaceFibre symbols and words, the user can also choose to trigger on specific errors.

The triggering capabilities of the SpaceFibre Link Analyser software currently match the functionality provided by the STAR Fire. This could be extended further in the future with the addition of further advanced hardware triggering capabilities.

VI. Symbol View

SpaceFibre uses 8B10B encoding to transfer 10-bit symbols over a SpaceFibre link. A symbol can be either a control or data symbol. A group of four consecutive symbols form a data word or control word.

The symbol view displays the captured SpaceFibre symbols and corresponding words travelling in both directions over a SpaceFibre link. One half of the symbol view displays the SpaceFibre traffic travelling in one direction whilst the other half displays the opposite direction. The left most column displays the time at which each word was captured relative to the trigger. For the STAR Fire unit this is simply the word index currently (the timing can be calculated knowing the link speed). For both SpaceFibre link directions there are four columns displaying four captured SpaceFibre symbols, plus a fifth column showing the SpaceFibre word the symbols equate to. Below is a screenshot of the symbol view showing data captured travelling in both directions.

Fig. 4. Symbol View

The SpaceFibre symbol and word types supported are those found in the latest draft of the SpaceFibre standard (SpaceFibre ECSS Draft H6 [1]). These have long been defined.

VII. Frame View

SpaceFibre uses frames to manage the flow of information over a SpaceFibre link. There are three frame types: data, broadcast and idle frames. Data frames are transmitted across a SpaceFibre link over virtual channels whilst broadcasts are transmitted over a broadcast channel. Idle frames are transmitted when there are no data or broadcast frames to be transmitted. Virtual channels provide multiple independent communication channels over a single physical link.

The frame view was designed to display the data and broadcast frames in their appropriate channel, relative to the capture trigger time. As with the symbol view, the left most column displays the time relative to the capture trigger. Every other column represents a virtual channel or broadcast channel as indicated by the column header. Below is a screenshot of the frame view showing data frames captured travelling in both directions over four virtual channels on a SpaceFibre link.

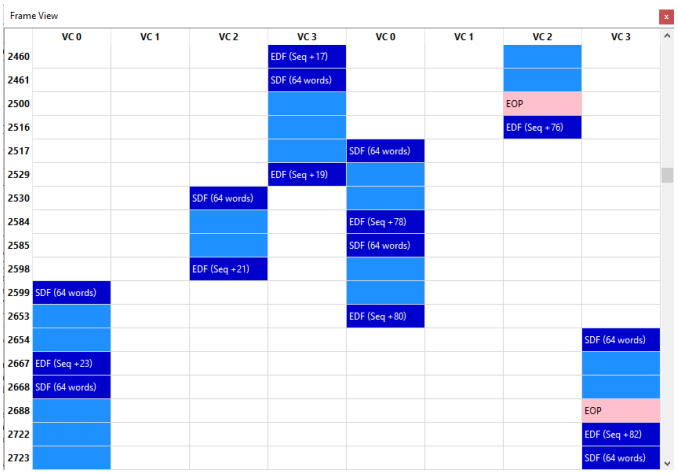


Fig. 5. Frame View

Each data frame consists of a start of a data frame (SDF) control word, up to 64 data words and an end of data frame (EDF) control word. Each broadcast frame consists of a start of broadcast frame (SBF) control word, two data words and an end of broadcast frame (EBF) control word. The properties of the different control words are displayed in both the symbol and frame views, the EDF word sequence number in the screenshot above is an example of this.

VIII. Packet View

A SpaceFibre packet consists of a destination address, cargo and an end of packet (EOP) or error end of packet (EEP) marker. The SpaceFibre packet format is the same as SpaceWire, enabling simple connection between existing SpaceWire equipment and high-speed SpaceFibre links. This also means existing software designed to display SpaceWire packets can be used as a basis for a SpaceFibre packet display.

One such view is the SpaceWire packet view provided with the SpaceWire Link Analyser Mk2 software application. The SpaceWire Link Analyser Mk2 packet view has benefitted from the feedback of numerous users over several years, and many users of SpaceWire, and potentially users of SpaceFibre, are familiar with this display. For these reasons it was decided that the SpaceWire Link Analyser Mk2 packet view should form the baseline design for the SpaceFibre Link Analyser software packet view. Revisiting the design provided an opportunity to extend the reusability of the view (to support packet data formats of multiple device types rather than simply one device) and review the functionality offered.

The SpaceFibre packet view displays the captured SpaceFibre packets travelling in both directions over a SpaceFibre link. The left most column displays the time relative to the capture trigger. Every other column represents a virtual channel. This display allows the user to view the SpaceFibre traffic at the packet level without concerning themselves with the symbols, words and frames used to construct the packets. Work on this view is currently ongoing and should be complete in the near future.

IX. Statistics

In addition to the SpaceFibre Link Analyser software, the STAR Fire will also be supplied with a standalone STAR Fire Statistics application. This displays virtual channel and broadcast channel statistics associated with the STAR Fire SpaceFibre data/broadcast generators and checkers. A running count of data errors, EEPs and broadcast errors is displayed alongside the data generator rate and bandwidth reservation for each channel. Virtual channel lane utilisation is also graphed over time. Below is a screenshot of the STAR Fire statistics application.

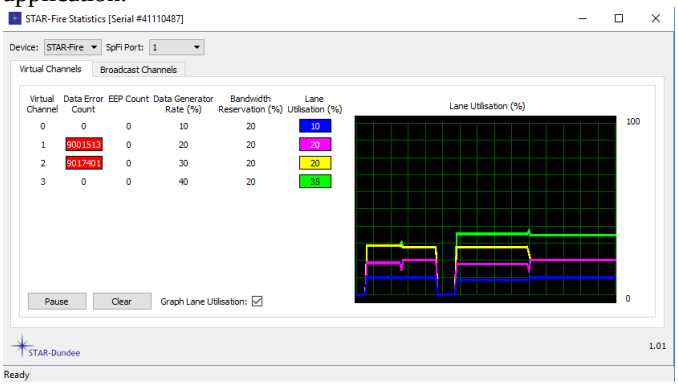


Fig. 6. STAR Fire Statistics

This immediately alerts the user to any errors detected by the data and broadcast checkers. The lane utilisation graph can be used to visualise the effect that changing the quality of service properties and data generator rate for each virtual channel has on lane utilisation.

X. Features

The symbol, frame and packet views provide a great way of inspecting captured SpaceFibre traffic at different levels of detail. Each view is docked within a separate floating window. The positioning of these windows is user configurable. They can be placed side by side, above and below, or on top of each other in separate tabs. This allows the user to layout the views in the most effective manner for them.

Selection of SpaceFibre traffic in one view automatically selects and navigates to the corresponding traffic in the other views. This allows the user to navigate multiple views simultaneously and therefore makes them much easier to manage.

Each view shall have search capabilities specific to that view. This will allow the user to quickly locate SpaceFibre traffic of interest within very large quantities of data that could otherwise be difficult and time consuming to identify. Filtering options shall allow users to limit the traffic presented to only that pertinent.

XI. Testing

Testing the SpaceFibre Link Analyser software was necessary throughout the development to ensure it behaved as expected. To test the SpaceFibre Link Analyser two SpaceFibre nodes were simulated transmitting and receiving SpaceFibre traffic to and from each other over a point to point link. This was achieved using a STAR Fire ("STAR Fire N"). A second STAR Fire unit ("STAR Fire LA") was inserted on the link and configured to operate as a SpaceFibre Link Analyser. The diagram below shows the setup.

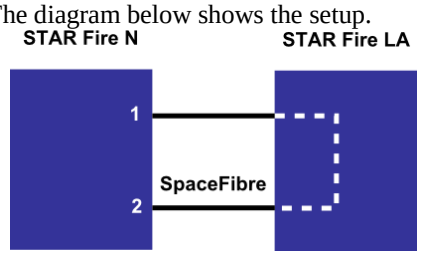


Fig. 7. SpaceFibre Link Analyser Test Setup

To configure STAR Fire N to transmit and receive SpaceFibre traffic the STAR Fire Controller application was used. The SpaceFibre port settings, quality of service, and advanced data generators and checkers can all be configured using the STAR Fire Controller. Numerous different configurations were used. Each could be saved and reused at a later date for regression testing. The STAR Fire Controller screenshot below shows virtual channel one settings for SpaceFibre port one of a STAR Fire unit.

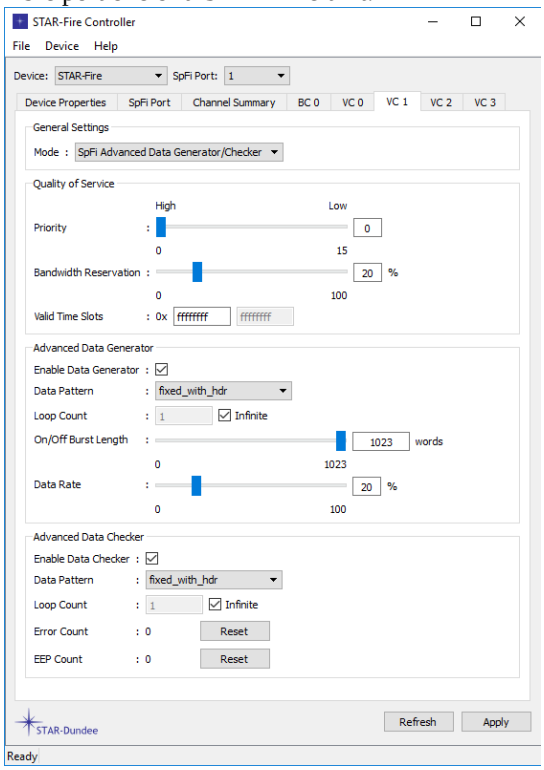


Fig. 8. STAR Fire Controller

The STAR Fire Controller application leverages the advanced data generators and checkers that have been added to the STAR Fire. These allow the user to specify data patterns of different types (fixed, increment, rotate right and rotate left), with an initial value, pattern length, packet length and four configurable header bytes. Below is a screenshot of the STAR Fire Controller dialog used to create and edit data patterns.

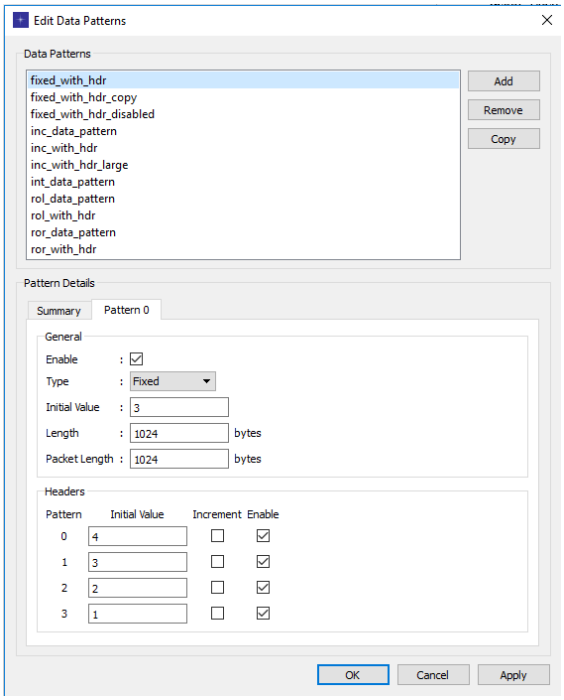


Fig. 9. STAR Fire Controller Advanced Data Patterns

XII. SpaceFibre Traffic Scenarios

The test setup described was used to capture and analyse a range of typical scenarios encountered on a SpaceFibre link. Below are some typical SpaceFibre scenarios captured by the STAR Fire and displayed using the SpaceFibre Link Analyser software.

A. Lane Initialisation

Lane initialisation is responsible for initialising a lane prior to transfer of data frames, idle frames or broadcast frames. This is handled by a lane state machine. A handshake protocol is used to ensure that both ends of the lane have achieved synchronisation. Below is a screenshot of the symbol view showing part of the lane initialisation handshake where the near end is moving to the connected state.

Fig. 10. Lane Initialisation Handshake

B. Frame Acknowledgment

Each correctly received SpaceFibre frame or FCT (Flow Control Token) is acknowledged with an ACK (acknowledgement) control word. Below are two combined screenshots of the symbol view showing a data frame with sequence number +47 captured travelling in one direction and the corresponding +47 ACK travelling in the opposite direction.

Fig. 11. Data Frame Acknowledgement

C. Error Recovery

A negative acknowledgement (NACK) is used to indicate that a data frame, broadcast frame or FCT has not been received correctly. NACKs are used to support link error recovery. In the symbol view screenshot below a NACK is captured indicating that a data frame, broadcast frame or FCT has not been received correctly.

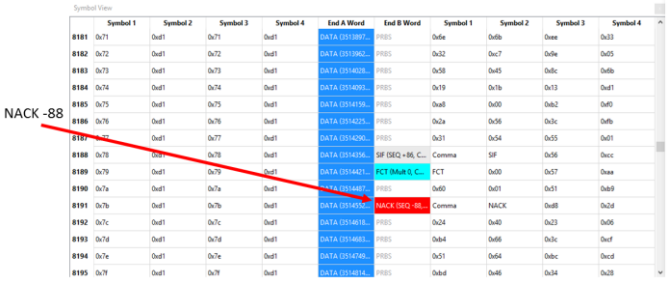


Fig. 12. Symbol View: Negative Acknowledgement

When the NACK is received this initiates the error recovery operation. The frame view screenshot below shows the retransmission of data frame 89. Data frame 89 was retransmitted as the NACK indicated that 88 was the sequence number of the last successfully received data frame, broadcast frame or FCT.

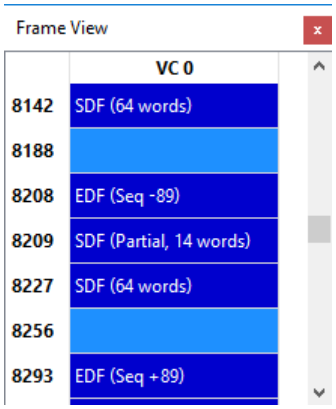


Fig. 13. Frame View: Data Frame Retransmission

D. Frame Precedence

SpaceFibre includes low latency event signalling and time distribution with broadcast messages. Broadcast frames have higher precedence than data frames ensuring broadcasts have minimum latency. Data frames have greater precedence than idle frames. Below is a screenshot of the symbol and frame views showing a broadcast frame embedded within a data frame.

Fig. 14. Broadcast Frame Embedded in a Data Frame

The data frame was part way through being transmitted when the broadcast frame became ready to send. The transmission of the data frame was suspended and the broadcast frame was sent immediately. After the broadcast frame was transmitted, the remainder of the data frame was transmitted.

E. Quality of Service Example

Each virtual channel is assigned quality of service (QoS) parameters that are used to determine which channel should be permitted to transmit data at any one time. Priority, bandwidth reservation and scheduling are used to do this. Below is a screenshot of the frame view showing four virtual channels where VC 0 is assigned 60% of the bandwidth, VC 1 10%, VC 2 10% and VC 3 10%. As you can see VC 0 is utilising the link far more than the other virtual channels as a result.

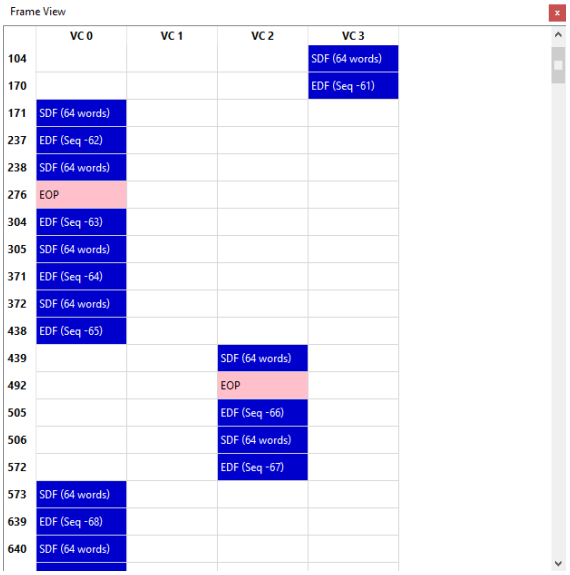


Fig. 15. Frame View QoS Example

XIII. Conclusion

As the popularity of SpaceFibre increases so too will the demand to perform effective SpaceFibre link analysis. This paper has described the current capabilities of the SpaceFibre Link Analyser software in an effort to make those responsible for the design and implementation of SpaceFibre networks aware of the existing tools available to them.

Currently in conjunction with the STAR Fire unit, users can capture SpaceFibre traffic in response to a trigger event. Once captured this traffic is automatically translated and displayed in symbol, frame and packet views. These views are all selection synchronised for easy navigation of the data. In the future additional functionality will be added to the existing views and additional devices will be supported by the SpaceFibre Link Analyser software.

References

[1] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements and M. Suess, ECSS SpaceFibre Standard Draft H6

[2] S. Parkes, https://www.star-dundee.com/spacefibre-users-guide, SpaceFibre User's Guide, STAR-Dundee Website

[3] STAR-Dundee, https://www.star-dundee.com/sites/default/files/STAR%20Fire_0.pdf, STAR Fire Data Sheet, STAR-Dundee Website

Networks & Protocols (Short)

Implementation and Validation of the SpaceWire-R Protocol

SpaceWire Networks and Protocols, Short Paper

Wojciech Mich
Krzysztof Romanowski
Piotr Tyczka
ITTI Sp. z o.o.
Poznań, Poland
{Krzysztof.Romanowski,Wojciech.Mich,
Piotr.Tyczka}@itti.com.pl

Rafał Renk
Adam Mickiewicz University
Poznań, Poland
Rafal.Renk@amu.edu.pl

Vangelis D. Kollias
Nikos Pogkas
TELETEL SA
Athens, Greece
{V.Kollias, N.Pogkas}@teletel.eu

***Abstract*— The paper deals with the aspects of implementation and validation of the SpaceWire-R protocol as carried out in the ESA-funded project SpaceR. We give a brief overview of the SpaceWire-R protocol for providing reliable data transfer services over SpaceWire networks and describe the SpW-R protocol software implementation elaborated in the SpaceR project. The testing platform developed for validation of the protocol is presented, as well as preliminary SpW-R performance results obtained within the project.**

***Index Terms*—SpaceWire, network protocols, SpaceWire-R**

I. Introduction

Many spacecraft on-board applications process and compress information in complex ways before sending it over a SpaceWire network. In these cases, a failure affecting even a small amount of data may lead to a loss of significant information. SpaceWire-R [1] is a communications protocol that addresses the needs of such applications by providing reliable data transfer services over SpaceWire networks. In addition to support for transmission reliability (with acknowledgements and retransmissions), its functions include multiplexing, segmentation, flow control, and keep-alive heartbeat.

The SpaceWire-R protocol has been implemented and tested as software on a PC and SpaceCube2 platforms [2] as well as an IP core targeting the Microsemi RTAX2000S FPGAs [3]. A revised version of the draft protocol specification has been recently issued [4]. As a step towards its standardization, an independent software implementation of the protocol based on the TELETEL iSAFT PVS platform [5] is under developement in the ESA-funded project SpaceR.

This paper presents the SpaceR project and its preliminary results. The objectives of the project include functional and performance tests and validation of the protocol, assessing its effectiveness, and deriving recommendations for the specification. The implementation is done in C++ on computers of x86_64 architecture and is portable at the source code level down to the layer of transfer of raw SpaceWire packets. The latter employs a new socket-based Application Programming Interface (API), developed by TELETEL specifically for the project. The testing environment is composed of a subsystem dedicated to upper-lower testing (for verifying the correctness of transformations between SpaceWire-R and SpaceWire packets) as well as one dedicated to end-to-end testing (for transmission between SpaceWire-R end point applications) with error injection facilities.

The paper is structured as follows. Section II gives an overview of the SpaceWire-R protocol and discusses the implementation aspects as realized in the SpaceR project. Next, in Section III a testing platform developed for SpaceWire-R testing and validation is described. The preliminary performance results obtained in the SpaceR project are presented and discussed in Section IV. Finally, Section V contains concluding remarks.

II. Implementation of the SpaceWire-R Protocol

The SpaceWire-R (SpW-R) communications protocol is intended to provide on-board applications with reliable data transfer services over SpaceWire networks. The position of the SpW-R protocol in the SpaceWire protocol stack is shown in Fig. 1. The main functions of the SpW-R protocol that are subject to implementation are the following: retransmission control, multiplexing, segmentation, flow control, and keep-alive.

A. Retransmission Control (reliable transfer)

- Transmits a series of data from the sender to the receiver without error, without loss, without duplication, and in sequence.
- Uses the concept of “transport channels”, which are virtual transmission lines from the sender to the receiver.
- A channel is established before starting transmission of data.
- Data is transmitted using SpW-R packets. A SpW-R packet is contained in a SpaceWire packet as its cargo. Each SpW-R packet is given a sequence number.

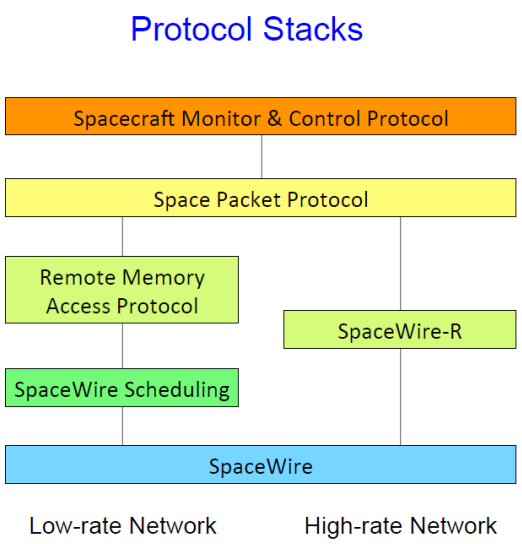


Fig. 1. SpaceWire-R protocol in the SpW protocol stack

- The receiver accepts packets in the order of the sequence number. When it has accepted a packet, it sends back an Ack (acknowledgment) packet to the sender.
- The sender retransmits the same packet if no Ack packet has been returned from the receiver. It disconnects the channel if no Ack packet has been returned after a certain number of retransmissions.

B. Multiplexing

- Enables simultaneous transfer of multiple independent streams of data from a sender node to a receiver node.
- Realized using multiple channels established between the sender and the receiver.

C. Segmentation

- Segments a data unit provided by the sending application into smaller segments so that each segment fits in a SpW-R packet, if the data unit provided by the sending application is larger than the size allowable in the SpW-R packet.
- At the receiver, the original data unit is reconstructed from the received segments and delivered to the receiving application.

D. Flow control

- Enables the receiver to dynamically inform the sender about how many more packets it can receive at the moment.
- Used if the receiver cannot accept many packets temporarily for some reason and wants the sender to slow down transmission.

E. Keep-alive (heartbeat)

- Enables detection of a line failure by sending signals between a sender and a receiver periodically when no data is being transmitted.
- When data is being transmitted, a line failure can be detected by not receiving Ack packets.

A SpW-R packet is sent as the cargo of a SpaceWire packet – preceded by the destination address and followed by the end-of-packet mark. The detailed structure of an SpW-R packet can be found in [4].

Key notions used to describe the protocol are the Transport End Point (TEP) and the transport channel. The TEP is defined as a point in a node that transmits (Tx TEP) or receives (Rx TEP) application data using a transport channel over a SpaceWire network. The transport channel is a protocol-defined one-way logical data path between a Tx TEP and a Rx TEP. There is only one transport channel between a Tx TEP and a Rx TEP; in other words, each TEP is dedicated to a certain channel. There can be multiple transport channels between any pair of nodes. This implies that multiple Tx and Rx TEPs can co-exist in a node. These relations are illustrated in Fig. 2.

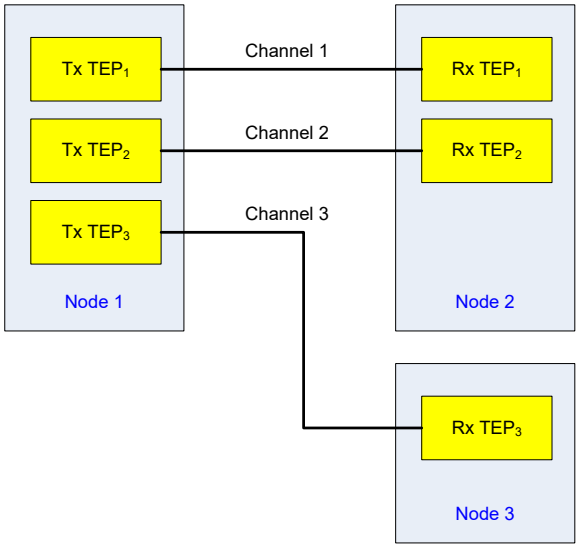


Fig. 2. Illustration of relationship between transport channels, TEPs, and nodes

The implementation of the protocol follows the relevant draft specification [4] and is intended to be separate from the rest of the system so as to be easily portable to possible other testing environments. As specified by the draft standard, the procedures performed by the protocol entities can be classified into three categories:

- procedures at a Tx TEP,
- procedures at a Rx TEP,
- common procedures at a node.

The TEP procedures communicate with an application sending or receiving the data in the form of Service Data Units (SDU). The protocol implementation provides function calls for the application to be used to request services and signals to inform the application of asynchronous events, e.g. a packet arriving from the network. These facilities are as follows:

- functions:
  - ChannelControl.request(ChannelNumber, DirectiveType)

- DataTransfer.request(ServiceDataUnit, SduId, ChannelNumber)

- signals:
  - ChannelControl.indication(ChannelNumber, NotificationType)
  - DataTransferNotify.indication(SduId, ChannelNumber, NotificationType, Reason)
  - DataTransfer.indication(ServiceDataUnit, TransportChannel)

It should be noted that according to [4] a number of protocol parameters related to transport channels are specified by means external to protocol service primitives or protocol packet data. With the implementation of the SpW-R protocol in the SpaceR project, the parameters are kept in a programmatic structure associated with the channel and can be set at runtime before or at invoking the operation of opening a transport channel.

The common procedures at a node communicate with the SpW network via the TELETEL's iSAFT PVS platform [5], specifically a new API to the iSAFT SpW Simulator, based on TCP sockets [6]. Thus the SpW-R to be transmitted via the network are embedded in SpW packets, which in turn are encapsulated in TCP segments and delivered to the iSAFT SpW Simulator.

The primary programming environment for the development of the SpW-R protocol implementation is C++ (gcc on Linux) with the standard library and some Boost libraries (notably MSM – the Meta State Machine – for implementing the finite state machine of the protocol, and Signals2 for implementing signals and slots).

III. Testing Platform

There are two groups of tests performed in the project:

- upper-lower and lower-upper tests, where the transformations between SpW-R and plain SpW packets, done by the protocol, are verified;
- end-to-end tests, where the correctness and performance of the protocol in actual traffic between end point applications is tested.

The underlying SpW network is a physical one, composed of one or more SpW switches. As the software is being implemented on general-purpose computers, an intermediary is required to connect to the SpW network; this is provided by the iSAFT PVS platform with a four-port SpW board. The board can be used for transmitting and receiving SpW traffic via the iSAFT SpW Simulator subsystem; detailed traffic and its statistics can also be observed and captured by the iSAFT SpW Recorder subsystem.

Since the iSAFT Simulator Client API allows one user application to be connected to the Simulator at the same time, in order to have more flexibility a Proxy system has been implemented, facilitating connection of several clients simultaneously, each of them having a dedicated SpW port (up to four total in the current configuration). The clients to the Proxy are mainly the implementations of the SpW-R nodes, which combine in a single compiled executable the SpW-R protocol implementation, the Proxy client, and the application producing or consuming the SDUs. The protocol implementation and the Proxy client are available as an object library.

In order to test the protocol behaviour in the presence of errors, a link emulator has been developed. This is an application that connects to the SpW network via two SpW ports, capturing plain SpW packets on either of them and optionally performing transformations on the packets before forwarding them through the other port. The transformations are directed by rules specifying under what conditions (triggers) a packet is to be transformed and what transformation (action) is to be done. Example triggers include the sequential number of the packet meeting certain criteria, as well as packet type, length, values of specific bytes. Possible actions include forwarding with no changes, dropping, delaying, truncating, extending or changing parts of the packet.

Figure 3 presents the various elements of the testing platform. Some of them exchange information inside the same executable; others communicate over TCP/IP. In order to minimize the overhead of the IP network, all the elements have been installed on the iSAFT PVS, in a dedicated virtual machine, although it is possible to run each block depicted on a separate computer.

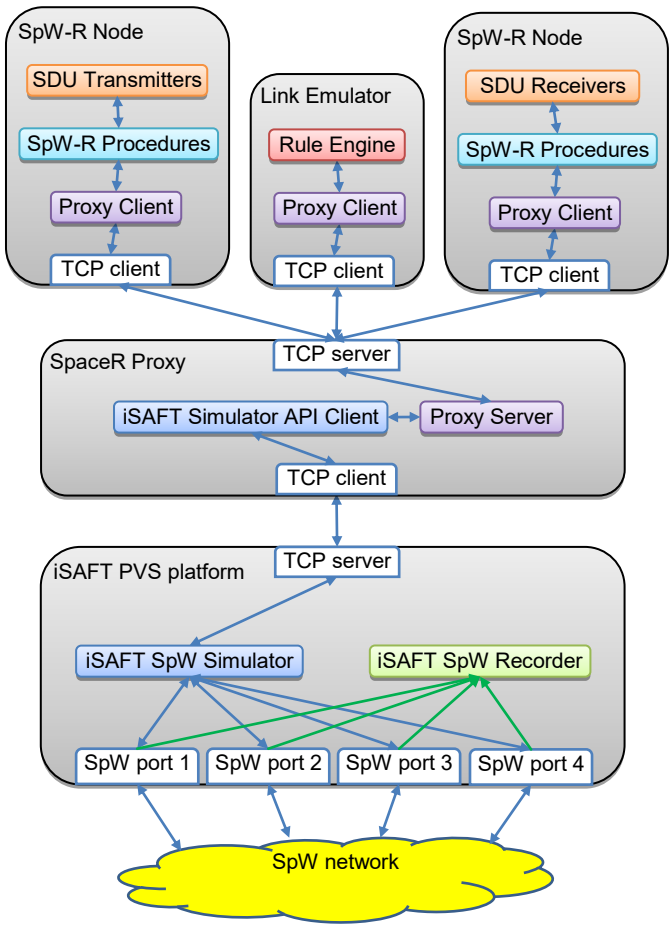


Fig. 3. Structure of the testing platform

Figure 4 shows the paths taken by data in an example configuration with two SpW-R nodes, each being used by three SDU transmitting/receiving applications, and a link emulator in the way between the nodes.

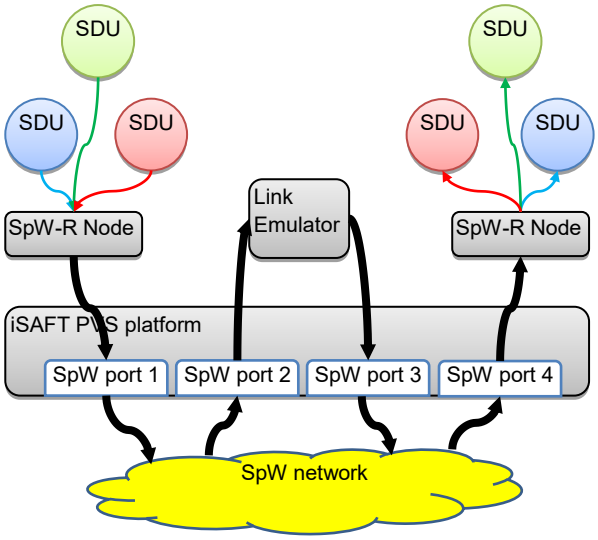


Fig. 4. Data flow through the testing platform

IV. Results

Preliminary tests have shown correctness of the protocol implementation, including boundary values for the protocol parameters. Comprehensive test runs are currently being executed, and the influence of a number of input parameters, including the SpW-R transport channel specifications, on the performance are examined. Figures 5 and 6 show some initial results for throughput and latency at the SDU level, the independent variables being the packet payload length (or SDU segment length) and the sliding window size. With the current implementation, obtaining throughput of the order of 50Mbit/s (as an objective of the project) with interfaces of 100Mbit/s line rate is feasible, although dependent on the sizes of the SDUs and segments.

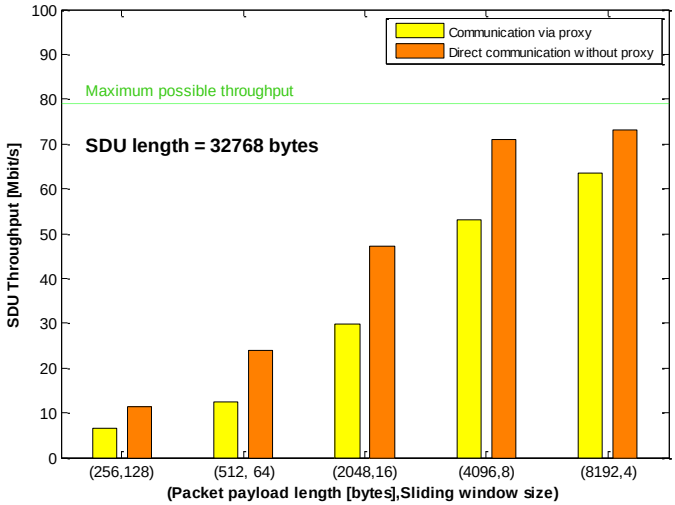


Fig. 5. Sample performance results obtained in the project (throughput)

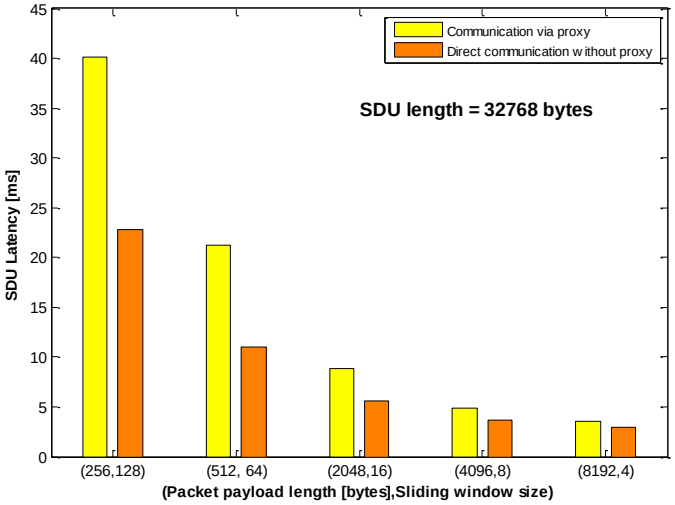


Fig. 6. Sample performance results obtained in the project (latency)

V. Conclusion

The paper discussed the implementation and testing of the SpaceWire-R protocol performed in the SpaceR project. The independent implementation is useful in standardization efforts. The testing platform, which is equipped with a flexible interactive operator's console involving full Python support and scripting, gives possibility of independent functional and performance testing.

Acknowledgment

This work has been funded by the European Space Agency under contract no. 4000112693/14/NL/CBi.

References

[1] T. Yamada, "Proposal for SpaceWire-R," 17th SpaceWire Working Group Meeting, ESTEC, Netherlands, December 2011

[2] K. Iwase, H. Hihara, O. Watanabe, T. Tanaka, T. Yamada, T. Yuasa, T. Tozawa, T. Tamura, "The evaluation of SpaceWire-R draft specification through the connectivity test using SpaceCube2," *Proc. 6th Int. SpaceWire Conf. Athens 2014*, pp. 82-85

[3] T. Yuasa, H. Hihara, T. Yamada, "SpaceWire-R core implementation and performance study," 24th SpaceWire Working Group Meeting, ESTEC, Netherlands, September 2015

[4] T. Yamada, "SpaceWire-R", issue 0.4 (draft), 13 August 2015

[5] A. Tavoularis, V. Vlagkoulis, N. Pogkas, V. Kollias, K. Marinis, "iSAFT-PVS: Recording, Simulation & Traffic Generation at Full Network Load," *Proc. 6th Int. SpaceWire Conf. Athens 2014*, pp. 73-79

[6] "iSAFT SpW Simulator. Client API Specification." TELETEL, 2016

SpaceWire Network Management Using Network Discovery and Configuration Protocol

SpaceWire Networks and Protocols, Short Paper

Krzysztof Romanowski[1], Piotr Tyczka[1], Witold Hołubowicz[2], Rafał Renk[2],
Vangelis D. Kollias[3], Nikos Pogkas[3], and David Jameux[4]

[1] ITTI Sp. z o.o., Poznań, Poland, {Krzysztof.Romanowski, Piotr.Tyczka}@itti.com.pl
[2] Adam Mickiewicz University, Poznań, Poland, {holub, Rafal.Renk}@amu.edu.pl
[3] TELETEL SA, Athens, Greece, {V.Kollias, N.Pogkas}@teletel.eu
[4] On-Board Data Systems (TEC-ED), ESA/ESTEC, Noordwijk, The Netherlands, David.Jameux@esa.int

***Abstract*— This paper presents the main outcomes of the ESA funded project entitled: "SPACEMAN – A SpaceWire Network Management Tool" that was jointly carried out by ITTI Sp. z o.o. and TELETEL SA., and dealt with SpaceWire Network Discovery and Configuration Protocol (SpW-NDCP). We give a brief overview of the SpW-NDCP, present the main features of the SPACEMAN tool for discovering and configuring SpaceWire networks, and introduce a novel XML representation of SpaceWire networks, implemented in the tool.**

***Index Terms*—SpaceWire, network management, NDCP, digital representation of SpaceWire networks.**

I. Introduction

It is well known that SpaceWire – a standard for high speed data-handling networks on board spacecraft [1, 2] – does not offer mechanisms for automatically discovering the topology of a network and devices (switches and nodes) it consists of. Moreover, services for configuring network devices and links are not present in the SpaceWire either. On the other hand, the increasing complexity and functionality of SpaceWire networks yield apparent need for providing such network management mechanisms.

To fill this gap, the SpaceWire Network Discovery and Configuration Protocol (SpW-NDCP) has been recently elaborated [3] as a development of the so-called "SpaceWire Plug-and-Play protocol" [4]. The SpW-NDCP assumes a simple two-layered architecture for performing network management activities that consists of a network management service and a communications protocol.

This paper presents the main outcomes of the ESA funded project entitled: "SPACEMAN – A SpaceWire Network Management Tool" that was jointly carried out by ITTI Sp. z o.o. and TELETEL SA. The general objective of the activity was to make a further development in the domain of SpaceWire network management through:

- developing a software implementation of the SpW-NDCP,
- developing a network management tool for discovering and configuring SpaceWire networks, and
- designing an XML representation of SpaceWire networks.

The paper reviews the SpaceWire NDCP and presents the basic features of the SPACEMAN Network Management Tool with the emphasis on functionality of design, discovery, comparison, and configuration of SpaceWire networks. The tool can handle networks with SpaceWire devices of different classes: NDCP-enabled, RMAP-enabled, and non-configurable. Methods for discovering them are shown in detail. Special cases of multiple control devices discovering the same network are also discussed. Finally, an advanced XML representation of SpaceWire networks, elaborated and used in the SPACEMAN project, is presented.

II. SpaceWire NDCP

The SpaceWire NDCP is one of the protocols that work over SpaceWire and is intended to permit SpaceWire network discovery and configuration in a standard and interoperable manner [3].

The SpW-NDCP protocol considers the SpaceWire network from the perspective of SpaceWire-based protocols. Applications are understood as users of the SpaceWire-based protocols and are the ultimate sources and destinations of messages carried over SpaceWire. In order to communicate over SpaceWire, each application uses a set of communication protocols, with SpaceWire itself as the lowest level protocol. For example, the Bepi Colombo and Solar Orbiter payload TM/TC is based on PUS [5] messages carried over SpW-CPTP [6], which itself is based on SpW-PID [7] over SpaceWire.

In NDCP terminology, every SpW End-Point and SpW Switch is referred to as an *NDCP Device* or *SpW Device* or *Device*. There are two kinds of NDCP Devices:

- Devices which will be managed by other Devices on the network are referred to as *Peripheral Devices*;
- SpW Nodes which will be engaged in managing Peripheral Devices on the network are referred to as *Control Devices.*

Devices are the functional elements of a SpaceWire network. Hence, the physical units of which a SpaceWire network is comprised may each be composed of one or more Devices.

SpW-NDCP assumes that each application and protocol has a number of management parameters which provide information on, or control, its operation. As each application uses one or more protocols, there are also management parameters which define an application's use of a protocol. The SpW-NDCP protocol provides a generic mechanism to access management parameters across a network.

The core set of NDCP management parameters offered by every Peripheral Device are those that relate to the Device itself. They allow the following to be identified:

1. the type of the device (i.e. whether it is a SpW Node or a SpW Switch);
2. the model of the Device (i.e. what product the Device is);
3. the physical unit that the Device belongs to;
4. the connections a Device has, enabling network discovery;
5. the protocols a Device supports;
6. the applications a Device supports;
7. the protocols (amongst the supported ones) that each application uses.

It is also possible to assign an identifier to the Device (Device ID). Device ID is necessary for network discovery to detect network loops.

The network management architecture consists of two layers:

1. a network management service; and
2. a communication protocol.

The network management service on a Control Device carries out network discovery, Device identification, and Device management activities using a communications protocol. A Peripheral Device allows itself to be managed by providing management parameters, which may be accessed using the communications protocol. The NDCP Control and Peripheral Device reference architecture is shown in Fig. 1.

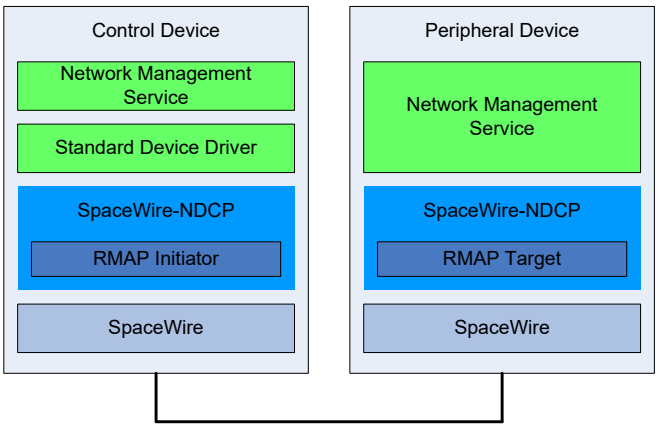


Fig. 1. SpW-NDCP Control and Peripheral Device reference architecture

The NDCP communications protocol provides a standard mechanism for accessing Peripheral Device management parameters from a Control Device. The communications protocol makes use of the syntax defined for the Remote Memory Access Protocol (RMAP) [8] and offers three operations: write, read, and compare-and-swap (CAS); not on memory addresses (like RMAP) but on NDCP Field Identifiers. Each operation accesses Peripheral Device information in a uniform way. Device information is held in regular sized fields of 32 bits and each field has an identifier. Related fields are grouped together into field sets. There are field sets for each supported protocol, service, and service protocol use. To identify a field as a part of read, write, or CAS operation, the Control Device must specify four values: the service (application) index, the protocol index, the field set identifier, and the field identifier.

III. SPACEMAN Network Management Tool

The primary functionality of the SPACEMAN Network Management Tool software application is discovering and configuring SpaceWire networks by employing features of the NDCP protocol. This functionality is complemented by facilities for editing and comparing network models, monitoring network changes, and exporting/importing internal network models to/from XML-format files. These functionalities are described below.

A. Network Discovery

The SpW Network discovery procedure builds up a graph model of the connected network by interrogating each Device in turn for the values of NDCP fields that it holds. These values describe the type of Device (Node vs. Switch), the number and status of its ports and other information. If the interrogated Device does not support the NDCP protocol, the interrogation is not answered and the SpW Network discovery application tries to read some information from the Device using the RMAP protocol, assuming that it might be a SpW-10X or a similar Device. If the RMAP reply values do not match certain assumptions adopted based on the SpW-10X memory layout and contents, or if there is no reply (but the relevant port status implies there is a device connected on the other end of the link), the device is assumed to be 'generic' and no further information on it is sought.

The application detects any loops that might be present in the topology of the network as well as any Device that might be discovered and owned by another instance of a management tool (another Control Device). A unique identifier is set on each discovered Device; with NDCP-capable devices, such an ID is protected from unauthorized overwriting by the NDCP protocol while, with SpW-10X devices that only support RMAP, the device 'identity' register is used by the SpW Network discovery application for ID storage, with no overwriting protection. Generic Devices do not have any discoverable identity and each time such a Device is encountered it is considered a distinct one. This does not constitute a severe limitation in case this Device is a SpW Node but could lead to undetected loops in the SpW Network in case this Device is a SpW Switch.

Figure 2 shows a graph of a discovered network. Windows listing the values read from two of the discovered devices are shown in Fig. 3 (for an NDCP-enabled Device) and Fig. 4 (for a SpW-10X device).

B. Network Configuration

Configuration consists of changing values of some NDCP fields stored in the Devices. This is possible either on a Device-by-Device basis by altering the values shown in the NDCP field windows (see Fig. 3); or by bulk loading values from an existing SpW Network model into all writeable NDCP fields of all Devices of the connected Network – provided the topology of the source model and the connected Network are compatible.

C. Network Model Editor

The application provides facilities for editing SpW Network models. A model can be created by the user from scratch or can be captured by discovering a physical Network. It is possible to add, modify, and delete Network model elements – SpW Nodes, Switches, and Links.

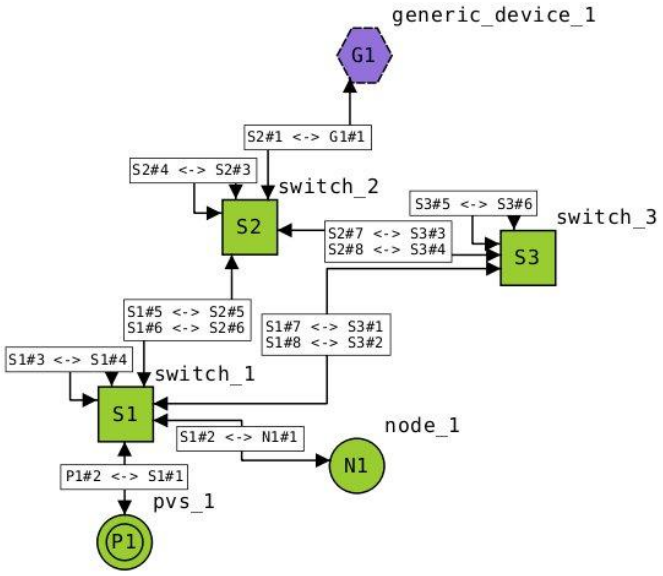


Fig. 2. Graph of a SpaceWire Network discovered by the SPACEMAN tool. Shown Devices are: square – Switch; circle – Node; double circle – Control Device; hexagon – generic SpW Device

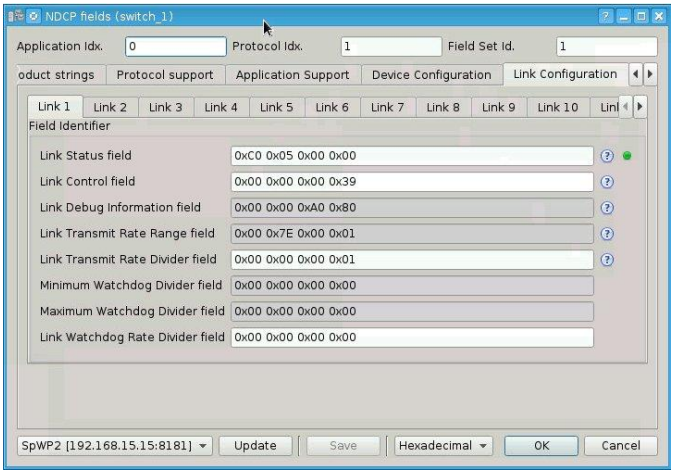


Fig. 3. NDCP field values read from a Device in the discovered Network

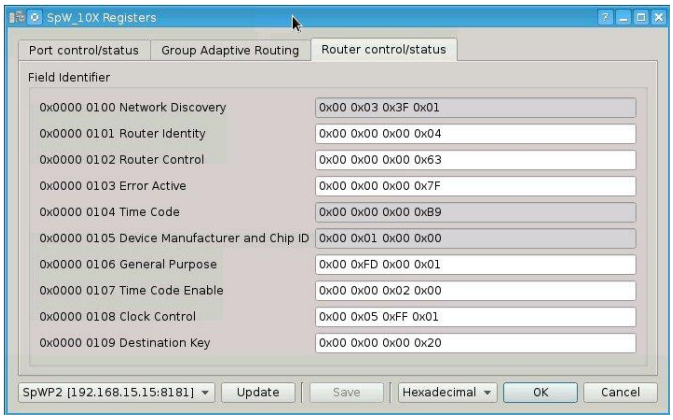


Fig. 4. SpW-10X registers read from a Control Device in the discovered Network

D. Network Comparison and Monitoring

Any two network models can be compared against each other and the differences marked by colour. Topological differences: an element of one model that is added to or missing from the other model, are marked in a graphical representation. Configuration differences: values of NDCP fields or SpW-10X registers different between the models, are marked in the graphical representation as changed Network elements and in the windows listing the values as changed fields.

The comparison can be invoked as a one-time operation on any two models, or it can be invoked automatically in a continuous discovery loop, in which the application repeatedly discovers the network and marks any changes found, thus monitoring its topology and configuration. Figure 5 shows a snapshot of a network monitoring session with some differences detected.

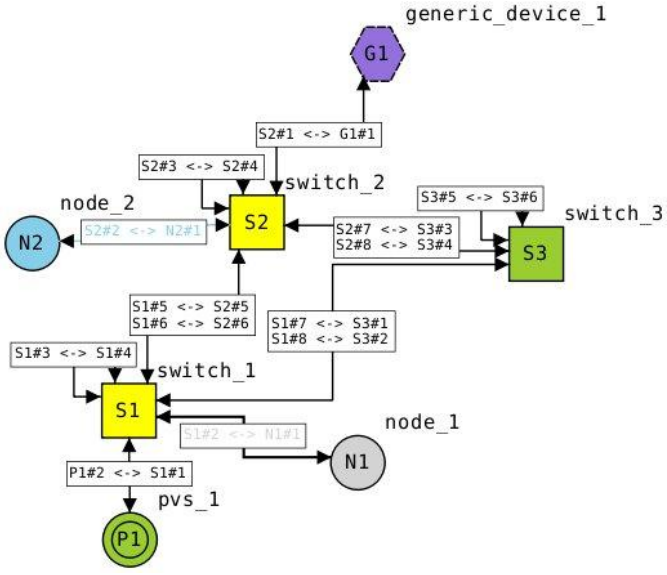


Fig. 5. Network differences discovered in a monitoring session. Element colours: blue – new; grey – missing; yellow – changed; green – unchanged

E. Packet-Level Testing

For trouble-shooting or fine analysis purposes, the Network Management Tool allows constructing, sending and receiving individual packets of different protocols: NDCP, RMAP, and plain SpaceWire, as well as matched command-reply transactions where applicable.

F. XML Export/Import

The internal SpW Network model can be exported to and imported from an XML file. The file to import from need not be written by SPACEMAN, as long as its format complies to the schema defined in SPACEMAN.

In its current version, the Network Management Tool connects to SpaceWire networks via TELETEL's iSAFT PVS hardware and Application Programming Interface [9], as shown in Fig. 6. The SpaceWire Network used in the project consisted of STAR-Dundee Mk2S Routers and Mk2 Bricks [10] (special versions with NDCP implemented in hardware) as well as a SpW-10X Router [11] with no NDCP support

.

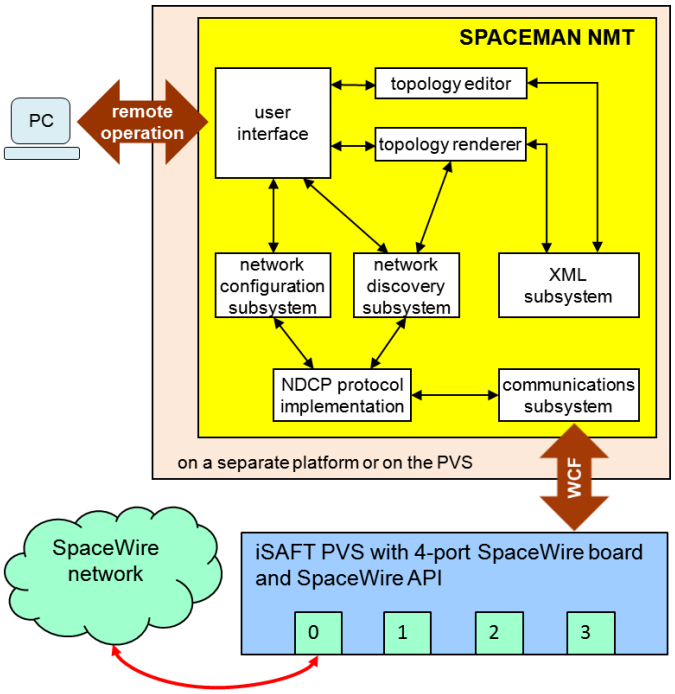


Fig. 6. Structure of the SPACEMAN Network Management Tool

IV. XML Representation of SpaceWire Networks

Developments of SpaceWire Networks naturally bore the need for representing these networks in a digital form. Digital representation is indispensable in many aspects related to SpW networks: designing, testing, preparing demonstrations, managing, etc. In recent years a few proposals for describing SpaceWire Networks by means of an Extensible Markup Language.(XML) format have appeared in the literature and been presented to the SpW community [6]-[8]. Elaboration of a holistic, universal structure for an XML representation of SpaceWire Networks is also of high interest to ESA/ESTEC. In reply to this need, the SPACEMAN project team made an attempt to develop a novel and advanced XML-based SpW Network representation. The proposed format is used in the XML export/import functionality of the tool (cf. Sec. III.F).

In the proposed XML structure, each element that represents a SpaceWire Device has some key attributes (e.g. name, number of SpW ports, logical address) and consists of a generic section and sections specific to the protocols that the Device supports. The generic section provides basic information on the device's ports. Both attributes and generic section are *redundant* in the sense that they contain information extracted from protocol sections. However, they provide the basic parameters of the device and hence favourably allow user to have a quick overview of these parameters. Protocol sections are made of nested sections and elements that correspond to the field sets, parameters, or registers relevant to a respective protocol.

The generic structure of the proposed XML scheme is depicted in Fig.7. The highest level element (root element), `DataHandlingSystem`, can contain several `Network` elements. Such an approach results from practical needs. For example, in real cases such as Bepi Colombo, several distinct SpW Networks are "linked" from data handling point of view by sharing common SpW Units. In the example shown in Fig. 7, a data handling system consists of only one SpW Network. Each `Network` element can contain the following nested elements that correspond to the Devices and Links the SpW Network is made of: `SpWSwitch`, `SpWNode`, and `SpWLink`. Additionally, when the NDCP protocol is used, the `NDCPControlDevice` element (or elements) is nested in the `Network` element as well. `SpWSwitch` and `SpWNode` elements have `NDCP_Device` attribute that specifies whether the device supports the NDCP protocol.

```
<?xml version="1.0" encoding="UTF-8"?>
<DataHandlingSystem Id="DHS1"
Name="DataHandlingSystem_1">
   <Network Id="Net1" Name="network_1">
      <NDCPControlDevice Name="pvs_1"
NoOfPorts="4" LogicalAddress="0xfc">
      <SpWSwitch Name="switch_101" NoOfPorts="8"
LogicalAddress="0xfe" NDCP_Device="false">
      <SpWSwitch Name="switch_102" NoOfPorts="8"
LogicalAddress="0xfe" NDCP_Device="true">
      <SpWNode Name="node_103" NoOfPorts="2"
LogicalAddress="0xfe" NDCP_Device="true">
      <SpWLink
Id="link_switch_101_4_switch_102_6">
   </Network>
</DataHandlingSystem>
```

Fig. 7. Example of the XML representation of a SpW network

An example of the XML representation of a SpW Switch that supports the NDCP is shown in Fig. 8. As can be seen from Fig. 8, the `SpWSwitch` element contains the generic section `<Ports>` and the section specific to the NDCP protocol `<NDCP>`. The `<NDCP>` section is made of sections and elements that correspond to the respective NDCP field sets. For non-NDCP Devices that support the RMAP protocol, this section is replaced by an `<RMAP>` section. The same structure

is used for SpWNode elements that describe SpW nodes in the XML representation. Examples of the XML representation of a SpW-10X and a SpW NDCP node are depicted in Figs. 9 and 10, respectively.

```
<SpWSwitch Name="switch_102" NoOfPorts="8"
LogicalAddress="0xfe" NDCP_Device="true">
   <Ports> ... </Ports>
   <NDCP>
        <DeviceIdentification> ...
        </DeviceIdentification>
        <VendorProductStrings> ...
        </VendorProductStrings>
        <ProtocolSupport> ...
        </ProtocolSupport>
        <ApplicationSupport> ...
        </ApplicationSupport>
        <DeviceConfiguration> ...
        </DeviceConfiguration>
        <LinkConfiguration NoOfLinks="12"> ...
        </LinkConfiguration>
        <SwitchingTable> ...
        </SwitchingTable>
        <TimeCodeGeneration> ...
        </TimeCodeGeneration>
        <ProtocolInformation> ...
        </ProtocolInformation>
   </NDCP>
</SpWSwitch>
```

Fig. 8. Example of the XML representation of a SpW NDCP switch (detailed description of sub-elements has been replaced by the ellipses for brevity)

```
<SpWSwitch Name="switch_101" NoOfPorts="8"
LogicalAddress="0xfe" NDCP_Device="false">
   <Ports> ... </Ports>
   <RMAP>
        <PortControlStatusRegisters> ...
        </PortControlStatusRegisters>
        <GroupAdaptiveRoutingTableRegisters> ...
        </GroupAdaptiveRoutingTableRegisters>
        <RouterControlStatusRegisters>
           <NetworkDiscoveryRegister Address="256"
   Value="0x00 0x03 0xF9 0x01"/>
           <RouterIdentityRegister Address="257"
   Value="0x00 0x00 0x00 0x65"/>
           <RouterControlRegister Address="258"
   Value="0x00 0x00 0x00 0x63"/>
           <ErrorActiveRegister Address="259"
   Value="0x00 0x00 0x01 0xF3"/>
           <TimeCodeRegister Address="260"
   Value="0x00 0x00 0x00 0x00"/>
           <DeviceManufacturerAndChipIDRegister
   Address="261" Value="0x00 0x01 0x00 0x00"/>
           <GeneralPurposeRegister Address="262"
   Value="0x00 0xFE 0x00 0x01"/>
           <TimeCodeEnableRegister Address="263"
   Value="0x00 0x00 0x02 0x00"/>
           <ClockControlRegister Address="264"
   Value="0x00 0x05 0xFF 0x01"/>
           <DestinationKeyRegister Address="265"
   Value="0x00 0x00 0x00 0x20"/>
        </RouterControlStatusRegisters>
   </RMAP>
</SpWSwitch>
```

Fig. 9.. Example of the XML representation of a SpW-10X (detailed description of sub-elements has been replaced by the ellipses for brevity)

```
<SpWNode Name="node_103" NoOfPorts="2"
LogicalAddress="0xfe" NDCP_Device="true">
   <Ports> ... </Ports>
   <NDCP>
        <DeviceIdentification> ...
        </DeviceIdentification>
        <VendorProductStrings> ...
        </VendorProductStrings>
        <ProtocolSupport> ...
        </ProtocolSupport>
        <ApplicationSupport> ...
        </ApplicationSupport>
        <DeviceConfiguration> ...
        </DeviceConfiguration>
        <LinkConfiguration NoOfLinks="4"> ...
        </LinkConfiguration>
        <TimeCodeGeneration> ...
        </TimeCodeGeneration>
        <ProtocolInformation> ...
        </ProtocolInformation>
   </NDCP>
</SpWNode>
```

Fig. 10. Example of the XML representation of a SpW NDCP node (detailed description of sub-elements has been replaced by the ellipses for brevity)

An example of a <Ports> section is given in Fig. 11. It contains elements that describe SpW ports of the Device. The attributes give information on the port number, the transmit rate (expressed in Mbit/s) and the status of a link associated with this port (connected/disconnected).

```
<Ports>
   <Port Number="1" TransmitRate="100"
PortConnected="true"/>
   <Port Number="2" TransmitRate="100"
PortConnected="false"/>
   <Port Number="3" TransmitRate="100"
PortConnected="false"/>
   <Port Number="4" TransmitRate="100"
PortConnected="true"/>
   <Port Number="5" TransmitRate="100"
PortConnected="true"/>
   <Port Number="6" TransmitRate="100"
PortConnected="true"/>
   <Port Number="7" TransmitRate="100"
PortConnected="true"/>
   <Port Number="8" TransmitRate="100"
PortConnected="true"/>
</Ports>
```

Fig. 11. Example of the <Ports> section

The structure of the <SpWLink> element is depicted in Fig. 12. The <SpWLink> describes a unidirectional link with the transmission direction from <Endpoint1> to <Endpoint2>. Hence, a bidirectional SpW link is represented by a pair of <SpWLink> elements.

```
<SpWLink Id="link_switch_101_4_switch_102_6">
   <Endpoint1 Name="switch_101" Port="4"/>
   <Endpoint2 Name="switch_102" Port="6"/>
</SpWLink>
```

Fig. 12. Example of the <SpWLink> element

The rules for the XML representation have been described in an XSD (XML Schema Definition) file.

V. Conclusions

The paper dealt with the issue of SpaceWire Networks management by means of the Network Discovery and Configuration Protocol. The topic was tackled from the perspective of the ESA-funded SPACEMAN project that aimed at developing a software network management tool (NMT) based on the NDCP for discovering and configuring SpaceWire networks.

The SPACEMAN NMT allows performing main NDCP operations, i.e. discovering unknown SpaceWire Networks and configuring SpaceWire Devices by changing values of some NDCP fields stored in the Devices. The discovery process is carried out for NDCP-enabled SpW Devices, as well as for SpW-10X or compatible devices that are not NDCP-enabled. The tool is also capable of discovering existence of so-called *Generic Devices* that are neither NDCP-enabled nor SpW-10X compatible. The tool includes some additional functionalities such as slow mode of discovery (step-wise), network comparison and monitoring, network model editing, packet-level testing for SpaceWire, RMAP and NDCP protocols, and SpW network model export to and import from XML files. As such, the SPACEMAN NMT allows testing and validating the NDCP protocol towards its future standardisation and can be a useful tool for engineers working with SpaceWire networks.

The novel digital representation of SpaceWire networks by means of XML was also presented in the paper. The approach aims at providing a holistic and flexible structure that is capable of representing a variety of different SpaceWire Networks and Devices. The proposed XML scheme is implemented in the SPACEMAN NMT.

Acknowledgment

This work has been funded by the European Space Agency under contract no. 4000109438/13/NL/Cbi.

References

[1] European Cooperation for Space Standardization, "Space engineering – SpaceWire – Links, nodes, routers and networks," ECSS-E-ST-50-12C, 31 July 2008

[2] European Cooperation for Space Standardization, "Space engineering – SpaceWire – Links, nodes, routers and networks," ECSS-E-ST-50-12C Rev.1 DIR3, 23 November 2015

[3] European Cooperation for Space Standardization, "Space engineering – SpaceWire Network Discovery & Configuration Protocol," ECSS-E-ST-50-54 Draft 1.7, 27 November 2014

[4] D. Jameux, "Towards SpaceWire Plug-and-Play ECSS standard," *4th Int. SpaceWire Conf.*, San Antonio, TX, USA, November 2011

[5] European Cooperation for Space Standardization, "Space engineering – Ground systems and operations – Telemetry and telecommand packet utilization," ECSS-E-70-41A, 30 January 2003

[6] European Cooperation for Space Standardization, "Space engineering – SpaceWire – CCSDS packet transfer protocol," ECSS-ST-E-50-53C, 5 February 2010

[7] European Cooperation for Space Standardization, "Space engineering – SpaceWire protocol identification," ECSS-ST-E-50-51C, 5 February 2010

[8] European Cooperation for Space Standardization, "Space engineering – SpaceWire – Remote memory access protocol," ECSS-E-ST-50-52C, 5 February 2010

[9] A. Tavoularis, V. Kollias, K. Marinis, "iSAFT Protocol Validation Platform for on-board data networks," DASIA Conference, Warsaw, 2014

[10] http://www.star-dundee.com

[11] C. McClements, S. Parkes, G. Kempf, "SpW-10X SpaceWire Router User Manual," Issue 3.5, 7 January 2015

[12] M. Takada, H. Takada, Y. Chen, M. Nomachi, T. Yuasa, T. Takahashi, "Development of Software Platform Supporting a Protocol for Guaranteeing the Real-Time Property of SpaceWire," *5th Int. SpaceWire Conf.*, Gothenburg, Sweden, June 2013

[13] T. Yuasa, T. Takahashi, M. Nomachi, "SpaceWire network XML representation. An example of JAXA-Nagoya University R&D project," *22nd SpW WG Meeting*, ESA/ESTEC, Noordwijk, The Netherlands, December 2014

[14] D. Raszhivin, "XML-based SpaceWire network representations for developments and network management," *22nd SpW WG Meeting*, ESA/ESTEC, Noordwijk, The Netherlands, December 2014

Constraint-based Configuration Table Generator for Reliable Path Routing and Safe Timeslot Allocation in SpaceWire Network

Session: Networks & Protocols, Short Paper

Satoshi Yamazaki, Toshio Tonouchi
System Platform Research Laboratories,
NEC Corporation
1753, Nakahara-ku shimonumabe, Kawasaki,
Kanagawa, 211-8666, Japan
s-yamazaki@bx.jp.nec.com, tonouchi@cw.jp.nec.com

Yu Otake, Yasuhiro Sota, Takahiko Tanaka
Space Systems Division,
NEC Corporation
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8501, Japan
y-otake@bp.jp.nec.com, sota@ab.jp.nec.com
t-tanaka@dy.jp.nec.com

Hiroki Hihara,
NEC Space Technologies, Ltd.
10, Nisshin-cho 1-chome, Fuchu, Tokyo, 183-8551, Japan
h-hihara@bc.jp.nec.com

Abstract—SpaceWire is valuable because it facilitates the development of spacecraft subsystems such as payload instruments, mass memory, and onboard computers. On the other hand, it takes much time and effort for developers to configure an initiator of the SpaceWire network because they have to take account of the entire SpaceWire network in a spacecraft. As the target network becomes larger, the path addressing and the packet collision-free timeslot allocation are harder for the developers to configure. Furthermore, the configuration tables of the initiator should satisfy various constraints, such as the bandwidth limitation and priority of specific packets. These constraints are different in each spacecraft. In order that the developers can design the large-scale SpaceWire network efficiently, automatic configuration table generation under the constraints is indispensable. This paper presents a constraint-based configuration table generator (CTG) that automatically provides reliable redundant path routing and collision-free timeslot allocation for required transactions in the target topology. We apply a constraint solver to the CTG to set many kinds of user-defined constraints in the network. For example, the bandwidth limitation, priority of the packets, and other various constraints can be easily inputted into the CTG. The CTG automatically generates configuration tables satisfying these constraints. Additionally, the CTG reports network topology views with bandwidth utilization ratios. This helps developers to verify whether a generated configuration is just as designed. The CTG can also notify developers that their requirements cannot be solved. In this paper, we show the feasibility and effectiveness of this tool through evaluation using a large-scale SpaceWire network case.

Index Terms— SpaceWire, SpaceWire-D, timeslot, constraint satisfaction problem, scheduling.

I. INTRODUCTION

SpaceWire has been adapted to a lot of satellite networks because it provides a standard data communication interface for spacecraft and facilitates the development of spacecrafts [1]. In ASTRO-H which is the space X-ray astrophysical observatory, all the transactions are controlled by the central master unit, which is called the satellite management unit (SMU) [2]. The transmission schedule of the transactions is described by configuration tables in the SMU. The SMU can realize deterministic transmissions of information, such as housekeeping data and instrumental data, by following the information in the tables.

On the other hand, it takes much time and effort for developers to test the functions of SMU. To test the configuration of initiators such as the SMU, developers have to take account of the entire network in a spacecraft. As the target becomes larger, the path address uses strict source routing in the SpaceWire network and the packet collision-free schedule is harder for the developers to configure.

In the actual SpaceWire network, all transactions are allocated to a timeslot introduced by SpaceWire-D to avoid packet collisions [3]. For example, the real time in ASTRO-H is divided into 64 timeslots using 64-Hz time codes emitted by SMU [2]. In addition, bus slots and system slots are introduced so that the SpaceWire network is able to operate in a deterministic manner without suffering from congestion and unexpected packet delivery delays. The important communications such as housekeeping data collections are allocated to bus slots while the system slots are used for the mission instrument data collections. However, it is too complex for developers to set the configurations that satisfy the user-defined management policy although such policies improve the reliability of the configurations.

Some authors have suggested scheduling methods for the SpaceWire network [4, 5, 6]. However, these methods cannot be adapted to ensure that the generated schedule satisfies user-

defined constraints such as timeslot number restriction, which depends on the kind of communication service. Hence, it is difficult to apply these methods to the actual SpaceWire network.

In this paper, we propose a constraint-based configuration table generator (CTG) that computes reliable path routing and safe timeslot allocation in the SpaceWire network. Communication requirements are inputted into the CTG, and the CTG outputs configuration tables that include routing tables for each initiator (initiator table) and tables for timeslot allocation (scheduling table). The CTG also outputs a network topology figure with bandwidth utilization information. This topology figure can clarify the bandwidth bottleneck in the target network and validate the outputted scheduling table.

II. Formulation

A. Architecture

CTG can make configuration tables of initiators in the target SpaceWire network. In a similar manner to that of related studies [5, 6], the configuration tables have information of transactions specified by the remote memory access protocol (RMAP), a standard communication protocol for the SpaceWire network [7]. To make the tables, network topology and communication requirement information are inputted into the CTG. In addition, constraint definition information, which describes user-defined requirements for timeslot allocation, is also inputted into the CTG. The details of the constraints are explained below.

The network topology information has the parameters of nodes, routers, and links in the target network. These parameters are the same ones used in preceding studies [5, 6]. The communication requirement information includes the parameters of the transactions. These parameters include the names of the source and destination nodes, RMAP address, RMAP packet types, and other pieces of information used for specifying communication service types.

By referring to the input information, the CTG searches all paths from the source node to the destination node for each required transaction. From all the paths, redundant paths are chosen to minimize the number of shared links. Hence, the CTG makes initiator tables that configure the redundant paths for each required transaction.

After making initiator tables, the CTG allocates timeslots to the required transactions. The details of timeslot allocation are shown in the section below. In the CTG, the scheduling tables for the target network are made in accordance with the result of timeslot allocation. By using the result, the CTG can also estimate the bandwidth utilizations of links for each timeslot. The bandwidth utilizations are visualized by generating network topology figures for each timeslot.

Note that the communication requirement with information of partially or fully allocated timeslots can be inputted into the CTG. The CTG can complement the timeslot allocation for remainder transactions if timeslots are partially allocated. In the case of fully allocated timeslot input, the CTG verifies whether the timeslot allocation satisfies the user-defined management policy.

B. Constraint-based timeslot allocation

The CTG solves the constraint satisfaction problem mentioned below to allocate timeslots to required transactions. Our goal is to find a timeslot allocation that satisfies all the constraints.

To control the allocation of transactions, we introduce Boolean decision variables as follows,

$$a_{m,n} \in \{0,1\}, \tag{1}$$

where $m = 0..N_t, n = 0..N_s - 1$. N_t and N_s are the number of required transactions and timeslots, respectively. The variable $a_{m,n}$ is set to 1 if transaction ID = m is allocated to the n th timeslot.

In this problem, we find an assignment of values of $a_{m,n}$ under the constraints below.

$$\forall m \sum_{n} a_{m,n} = \frac{N_s}{T^m_{period}}, \tag{2}$$

$$\forall l \sum_{(m,x)\in TRANS_l} \frac{T^m_x}{T_{slot}} a_{m,n} = 1.0, \tag{3}$$

$$\forall m \sum_{n \notin SLOTS_m} a_{m,n} = 0, \tag{4}$$

$$\forall n \sum_{m} a_{m,n} \le N^n_{max}, \tag{5}$$

$$minimize \sum_{l\in L} \left(\sum_{(m,x)\in TRANS_l} \frac{T^m_x}{T_{slot}} a_{m,n} \right)^2, \tag{6}$$

where T^m_{period} is the number allocated in 1 cycle timeslot for the m th transaction [6]. In the case of $N_s = T^m_{period} = 64$, constraint (2) ensures that there is only one decision variable set to 1 for each required m th transaction since the T^m_{period} set to 64 means the transaction is required to be used for each time slots.

Constraint (3) represents the bandwidth constraint ensuring that capacity limitations will not be exceeded. T_{slot} is the time for each timeslot. In the case of $N_s = 64$, $T_{slot} = 15.625\ ms$. The set of tuples, $TRANS_l$, shows all the IDs of transactions that pass the link . The ID of a transaction that has passed and the packet's direction are indicated by m and x, respectively. In a SpaceWire network, transactions include a sending packet (s) and a reply packet (r). The spending times of two packets are individually estimated in the CTG. The spending time T^m_x $(x = s, r)$ is evaluated with the following formulas, which are based on related studies [5, 6].

$$T^m_s = T^m_{srt} + T^m_{smdt}, \tag{7}$$

$$T^m_r = T^m_{rrt} + T^m_{rmdt}. \tag{8}$$

The time spent on sending the mth packet T^m_{srt} and the time spent on sending the reply to the mth packet T^m_{rrt} depend on the RMAP packet type of the mth transaction.

RMAP write:

$$T_{srt}^{m} = 10 \times \frac{(R^m + P^m + D^m + 17)}{S} + T_{pd}R^m \quad (9)$$

$$T_{rrt}^{m} = 10 \times \frac{(R^m + 8)}{S} + T_{pd}R^m, \quad (10)$$

RMAP read:

$$T_{srt}^{m} = 10 \times \frac{(R^m + P^m + 16)}{S} + T_{pd}R^m, \quad (11)$$

$$T_{rrt}^{m} = 10 \times \frac{(R^m + D^m + 13)}{S} + T_{pd}R^m, \quad (12)$$

RMAP read modify write:

$$T_{srt}^{m} = 10 \times \frac{(R^m + P^m + 25)}{S} + T_{pd}R^m, \quad (13)$$

$$T_{rrt}^{m} = 10 \times \frac{(R^m + 17)}{S} + T_{pd}R^m, \quad (14)$$

where D^m is the transmission data length of the m th transactions. The spending time is strongly affected by D^m. Hence, T_{srt}^{m} in the RMAP write packet is larger than T_{rrt}^{m}. The minimum line speed (M bit/s) in the target network is indicated by S. In the above formula, the spending time also depends on the number of routers that have been passed R^m, delay time of each router T_{pd}, and number of reply path addresses P^m. The spending times defined by the above formula are estimated by the data size of the packet header specified by SpaceWire [1, 7].

Constraint (4) and (5) reflect a user-defined constraint that is different in each project. For example, the transactions for housekeeping data are allocated to the $(4n + 0)$th timeslot, which is called the bus slot. This restriction can be encoded in constraint (4), where $SLOTS_m$ is the set of permitted timeslots for the ID = m transaction. The information of $SLOTS_m$ is included in the constraint definition information inputted into CTG. By using constraint (4), unpermitted timeslot allocation can be prohibited because the values of $a_{m,n}$ where $n \notin SLOTS_m$ become zero.

For another example, the limitation of the number of specific transactions for the bus slot is described by the network management policy. This policy can be encoded in constraint (5). Constraint (5) ensures that the number of transactions for the nth timeslot does not reach the maximum number N_{max}^{n} given by the inputted constraint definition information. Optionally, CTG can consider the optimization constraint (6), which disperses the bandwidth utilization of links.

Many existing off-the-shelf SMT solvers can solve the scheduling problem under constraint (2-5). In addition, Z3 [8], an existing SMT solver, can also consider optimization constraint (6).

III. Experiment

We apply the proposed CTG to a SpaceWire network that is composed of 1 initiator (BMC) and 3 targets (data recorder, Node 1, and Node 2). There are 70 transactions per second in this system. To show the function of timeslot allocation in the CTG comprehensively, the redundancy of all the paths is set to 1 in these evaluation results. The information inputted into the CTG is in XML format based on preceding studies [5, 6]. Figure 1 shows a snippet of the constraint definition information.

```
<SpaceWireConstraintsInfo>
  <PermittedSlot CharacterId="Housekeeping"
      Slot="0;4;8;12;16;20;24;28;32;36;40;44;48;52;56;60"/>
  <PermittedSlot CharacterId="System Polling"
    Slot="5;6;7;9;10;11;13;14;15;17;18;19;21;22;23;25;26;27;29;30;31;33;
        34;35;37;38;39;41;42;43;45;46;47;49;50;51;53;54;55;57;58;59"/>
  <Character Id="Housekeeping ">
    <Packet ServiceType=" Housekeeping" />
  </Character>
  <Character Id="System polling">
    <Packet ServiceType="System polling" />
  </Character>
</SpaceWireConstraintsInfo>
```

Fig. 1. Snippet of constraint definition information (XML format).

TABLE I. Result of Timeslot Allocation.

(a) Timeslot table.

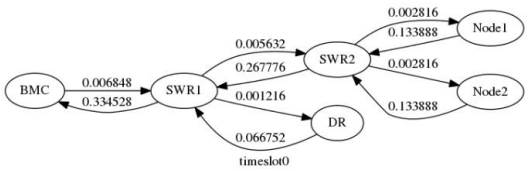
Slot number	Transaction IDs
0	0,1,4,5,8
1	
2	
3	
4	0,1,3,4,5,7,8
5	
6	2,6
7	9, 10

(b) List of parameters of required transactions.

ID	Source	Destination	Type	Service
0	BMC	Node 1	Read	Housekeeping
1	BMC	Node 1	Read	Bus polling
2	BMC	Node 1	Read	System polling
3	BMC	Node 1	Read	Instrument
4	BMC	Node 2	Read	Housekeeping
5	BMC	Node 2	Read	Bus polling
6	BMC	Node 2	Read	System polling
7	BMC	Node 2	Read	Instrument
8	BMC	DR	Read	Housekeeping
9	BMC	DR	Write	Command
10	BMC	DR	Write	Command

The results of timeslot allocation for the 0th - 7th slot are shown in Table 1. Note that the transactions in this experiment are allocated to 64 timeslots ($N_s = 64$). The allocated transaction IDs for each timeslot are shown in Table 1(a). Table 1(b) shows the parameters of the transactions. In this experiment, we consider the constraints that bus polling and housekeeping communications have to be allocated to the bus slot. Therefore, ID = 0, 1, 4, 5, 8 are allocated to slot 0 or slot 4

(a)



(b)

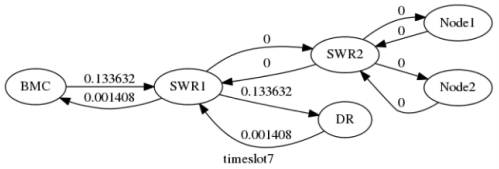
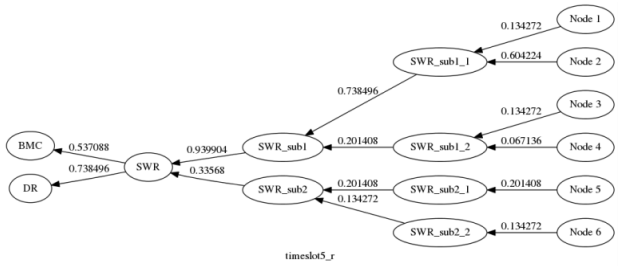


Fig. 2. Network topology figures for case of 1 initiator (BMC) and 3 targets (DR, Node 1, and Node 2). DR stands for data recorder. (a) 0th slot (b) 7th slot. The figures show bandwidth utilizations of sending and reply packets by each link.

(a)



(b)

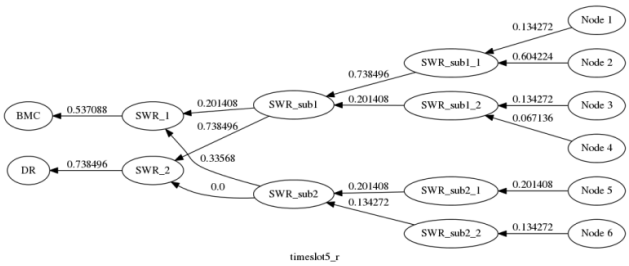


Fig. 3. Network topology figures for case of 2 initiators (BMC, DR) and 6 targets (Nodes 1-6). Bandwidth utilizations of reply packets are shown.

in Table 1(a) because the results of the CTG always satisfy the inputted user-defined constraints such as the restriction shown in Fig. 1. The network topology figures for the 0th slot and 7th slot are shown in Fig. 1. The circles and arrows on the maps are network nodes and links, respectively. The maps also show bandwidth utilizations by each link.

In the figures, arrows from the left nodes to the right nodes indicate sending packets; arrows in the opposite direction indicate reply packets. Figure 2(a) shows that the bandwidth utilizations of reply packets are larger than those of the sending packets because the RMAP packet types of all transactions allocated to the 0th timeslot are RMAP read transactions. On the other hand, bandwidth utilizations of the sending packets are larger in Fig. 2(b) because the RMAP packet types of all transactions allocated to the 7th timeslot are RMAP write transactions.

Figure 3 shows the results of another experiment where the evaluated network is composed of 2 initiators (BMC, DR) and 6 targets (Nodes 1-6). The experiment simulates timeslot allocation for 120 transactions for data collection. Figure 3(a) shows that the bandwidth utilization ratios for all links are lower than 100% because of the safe timeslot allocation by using the CTG. As a result, the link between SWR and SWR_sub1 becomes the bandwidth bottleneck in the network. This bottleneck seems to be reduced by adding another router. Figure 3(b) shows the results of the additional experiment. The visualization of bandwidth utilization is useful for developers to make a strategy for increasing the total bandwidth utilization efficiently.

IV. Conclusion

In this paper, we propose a constraint-based configuration table generator to reduce the number of man-hours for verification of network configurations. The configuration tables outputted by the CTG can be applied to an actual SMU since the tables satisfy the user-defined constraints encoded from management policies for each project.

References

[1] ECSS-E-ST-50-12C, “SpaceWire - Links, Nodes, Routers and Networks” 31 July 2008.

[2] T. Yuasa, T. Takahashi, M. Ozaki, and M. Kokubun, “A deterministic SpaceWire network onboard the ASTRO-H space x-ray observatory,” in International SpaceWire Conference, Texas, USA, November 2011.

[3] S. Parkes, A. Ferrer, S. Mills, and A. Mason, “Spacewire-D: Deterministic data delivery with SpaceWire,” in International SpaceWire Conference, St Petersburg, Russia, June 2010.

[4] D. Raszhivin, Y. Sheynin, and A. Abramov, “Deterministic scheduling of SpaceWire data streams,” in International SpaceWire Conference, Gothenburg, Sweden, June 2013.

[5] Y. Chen, M. Takada, R. Kurachi, and H. Takada, “A scheduling method of RMAP packets for SpaceWire-D,” in International SpaceWire Conference, Gothenburg, Sweden, June 2013.

[6] M. Takada, H. Takada, Y. Chen, T. Yuasa, T. Takahashi, and M. Nomachi, “Development of software platform supporting a protocol for guaranteeing the real-time property of SpaceWire,” in International SpaceWire Conference, Gothenburg, Sweden, June 2013.

[7] ECSS-E-ST-50-52C, “SpaceWire - Remote memory access protocol” 5 February 2010.

[8] L. De Moura and N. Bjørner, “Z3: An efficient SMT solver,” in International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2008, pp. 337–340.

Essential SpaceWire Hardware Capabilities for a Robust Network

Session: SpaceWire Networks and Protocols, Short Paper

Michael Birmingham
NASA Goddard Space Flight Center/ASRC Federal
Denver, CO USA
E-mail: mike.j.birmingham@lmco.com

William H. Anderson
NASA Goddard Space Flight Center/ASRC Federal
Greenbelt, MD USA
Email: william.h.anderson@nasa.gov

Alexander Krimchansky
NASA Goddard Space Flight Center
Greenbelt, MD USA
Email: alexander.krimchansky@nasa.gov

Matthew S. Lombardi
Lockheed Martin
Denver, CO USA
Email: matthew.s.lombardi@lmco.com

***Abstract*— The Geostationary Operational Environmental Satellite R-Series Program (GOES-R) mission is a joint program between National Oceanic & Atmospheric Administration (NOAA) and National Aeronautics & Space Administration (NASA) Goddard Space Flight Center (GSFC). GOES-R project selected SpaceWire as the best solution to satisfy the desire for simple and flexible instrument to spacecraft command and telemetry communications. GOES-R development and integration is complete and the observatory is scheduled for launch October 2016.**

The spacecraft design was required to support redundant SpaceWire links for each instrument side, as well as to route the fewest number of connections through a Slip Ring Assembly necessary to support Solar pointing instruments. The final design utilized two different router designs.

The SpaceWire standard alone does not ensure the most practical or reliable network. On GOES-R a few key hardware capabilities were identified that merit serious consideration for future designs. Primarily these capabilities address persistent port stalls and the prevention of receive buffer overflows. Workarounds were necessary to overcome shortcomings that could be avoided in future designs if they utilize the capabilities, discussed in this paper, above and beyond the requirements of the SpaceWire standard.

***Index Terms*—SpaceWire, Networks, Routers, GOES-R, GRDDP.**

I. Introduction

This paper seeks to describe some of the pitfalls encountered during the design and integration of major components for the Geostationary Operational Environmental Satellite-R Series (GOES-R) program [1]. An awareness of those pitfalls may prevent a similar experience in future designs.

The GOES-R spacecraft uses European Cooperation for Space Standardization (ECSS) SpaceWire [2] for the transfer of sensor, telemetry, ancillary, command, time code, and time synchronization information between instruments and the spacecraft. Capabilities beyond those specified in the standard are offered in the interest of providing a more robust system.

This paper describes four instances where considerable effort was expended to avoid or mitigate problems concerning persistent port stall, receive buffer overflow, pipeline side-effects, and a situation where buffer depth configuration of a node exposed a router defect that locked out further transfers. This specific configuration can be avoided given the details in that section.

A. Background Information

GOES-R uses Reliable Data Delivery Protocol (GRDDP [3]) which specifies that Reset packets are transmitted at that channel's transmit timeout rate from the time that the channel is placed into an Enabled state, until an Acknowledge packet is received. The transmit timeout is specified in an instrument Interface Control Document (ICD), and is on the order of 100ms for the instruments described in this paper. During instrument power-on, the spacecraft will begin transmitting Reset packets (9 bytes in length) to the instrument at a 100ms rate until the instrument responds.

The spacecraft transition to Enabled state is delayed from the application of instrument power to coincide with the point at which the instrument enters Run Mode, and is able to process GRDDP messages. If the instrument indeed enters Run Mode at about the expected time, few Reset packets will be transmitted. Problems may arise, however, if there is a problem with either the instrument or the link.

GOES-R also specifies that instruments shall transition to a Safe Mode if time ticks or time-of-day messages are absent for 10 consecutive seconds.

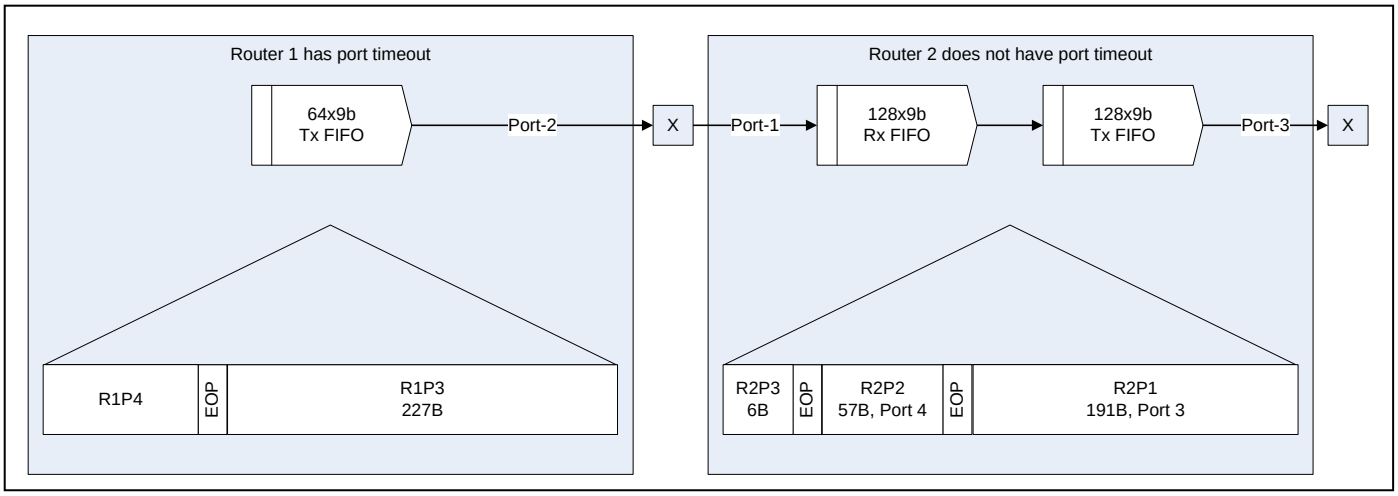


Figure 1 Example Routers and Packets for Transmit Timeout Discussion

I. Persistent port Stall Prevention

The first capability to be discussed is a mechanism to prevent an indefinite network stall. This is especially important when routers are employed between nodes. The need will be illustrated in the following sections by way of an example.

A. Routing example

The following is a simplified example of a real-world condition encountered on GOES-R during instrument emulator integration. Router 1 in the following example is implemented as a Goddard Space Flight Center (GSFC) developed core [4] which is part of the BAE Systems SpaceWire ASIC [5]. Router 2 is an Aeroflex 4-port router [6].

Details such as Lookup Tables (LUTs), registers, arbiters and other router components are not included since it assumed that the reader has a working knowledge of those mechanisms.

B. Initial Condition

In Figure 1 above, Router 1 has port transmit timeout capabilities, while Router 2 does not. Router 2 Port 3 is in disconnect due to an instrument or cable failure, and cannot reconnect. This condition may be present prior to instrument power-on or may occur during operation.

R2P1 is Router 2 Packet 1; it is 192 bytes including the End of Packet marker (EOP) and its destination is Port 3. No part of Packet 1 has been transmitted yet, in this example.

R2P2 is Router 2 Packet 2; it is 58 bytes including the EOP and will be routed to Port 4 (not shown).

R2P3 is the leading portion of Packet 3, while R1P3 is the trailing portion of Packet 3. Packet 3 is 234 bytes in total, including the EOP.

R1P4 is the final packet to be queued up for Router 1, but neither the length nor the EOP is indicated because it is not relevant for this example.

C. Stall Condition

R2P1 will not be delivered due to the disconnect condition on Router 2 Port 3. Since R2P1 exceeds the size of Port 3's transmit (Tx) First In First Out memory (FIFO), it will block Port 1's arbiter. The trailing portion of Packet 1 and all of Packet 2 will occupy all but 6 bytes of Port 1's Receive (Rx) FIFO. The remaining free space on Router 2's Port 1 Rx FIFO will be filled with the leading 6 bytes of Packet 3.

D. Timeout Condition

Router 1 Port 2 has not completed transmitting Packet 3 within the programmed timeout limit, and disconnects Port 2. Pursuant to ECSS error recovery specifications, Router 1 will spill the trailing 228 bytes of Packet 3. Router 2 will not append an EEP to partial packet 3 because there is no space in the receive buffer.

E. Link Recovery

Both Router 1 and Router 2 will issue NULL characters in an attempt to re-establish the link. Assuming Router 1 Port 2 Rx FIFO (not shown) has at least 8 bytes free, it will also issue one or more Flow Control Token (FCT) characters. Router 2, on the other hand, will not issue an FCT because there are no bytes free in its Rx FIFO.

The ECSS standard does not have a remedy for this situation. It is assumed that there are no hard link errors, and that eventually data will flow through Router 2 Port 3. If the failure is not recoverable with an instrument power cycle (if it can even be identified by the host system) then the failure will persist ad infinitum.

F. GOES-R Configuration

On GOES-R, only the first router in the chain is capable of disconnect on a transmit timeout, and it is not on a per-port basis; the timeout applied to all ports equally. The routers downstream (Aeroflex 4-port routers) of that router had no transmit timeout capability. The indefinite stall cannot be avoided unless all routers have the transmit timeout capability.

G. GOES-R Stall Consequences

The perpetual stall means that all instruments downstream of Router 2 Port 1 will be unable to communicate. Instrument telemetry will not be acknowledged, and instruments will no longer receive commands, time messages or time ticks. Within 10 seconds instruments fall into Safe Mode. All GOES-R GRDDP transmit channels to those instruments close and numerous error events result. Unless the condition was present during the power-on process, there is no way of knowing which port of which router was in disconnect.

H. GOES-R Recovery Method

The GOES-R recovery method begins by powering off all instruments downstream of Router 1 port 2. A hard reset is then required of the routers downstream of Router 1 (there are four on GOES-R). Each hard reset clears the FIFOs and all router registers are returned to default values. The reset does not affect the LUT contents. Next, the registers have to be re-configured for each router. As each instrument is powered up, their router port status is examined. If not in Run State, the instrument is swapped to the redundant side.

I. Recommended Design Solution

On any network involving one or more nodes, a programmable transmit timeout feature on **every** router port in the chain is essential to preventing a perpetual stall somewhere in the chain. Of course the timeout must apply to any packet that stalls the transmitter, even if the port is in disconnect and no part of the packet has been sent. The ECSS standard specifies only that a partial packet be spilled when the link error is reported (transition to disconnect).

The transmit timeout feature on all routers will clear the stall but as long as the point of origin continues transmitting packets to the node in disconnect then the behavior repeats indefinitely. Best practice would be to check port status prior to and following instrument power-on, as well as periodic monitoring.

There may be considerable packet jitter with this solution, caused by the timeout that must expire before a packet is spilled. When using GRDDP, the port timeout setting must be much less than the shortest re-transmit timeout, since Reset packets have priority over all but Ack packets and Reset packets will likely be prevalent in this situation.

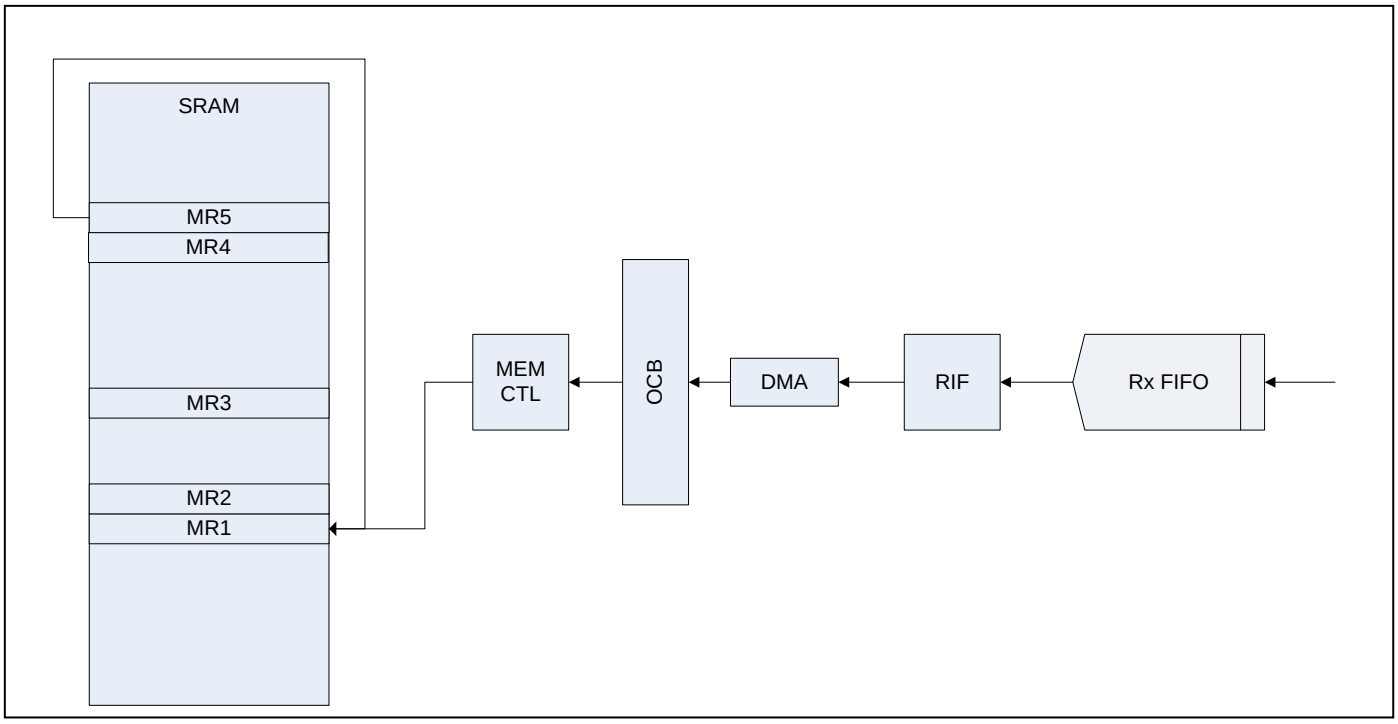


Figure 2 Example Receive Buffer Chain

II. Receive buffer overflow prevention

The next capability to be discussed is a mechanism to prevent receive buffer overflow. The ECSS standard does not limit the length of data packets, but practical applications should limit the size of packets, as does the GRDDP protocol. On GOES-R, each instrument also had constraints on the size of packets that were to be transmitted or received, which were equal to or less than what the protocol allowed.

The ECSS standard assumes that FCT messaging will prevent receive buffer overflow (section 8.3). In reality, receive FIFO overflow is prevented, but not necessarily receive buffers. Rx FIFOs are the domain of hardware and credit counts and FCT transmit is the purview of that lower level of the system. The practical problem is that the receive front end may have no idea of the size of the buffer in system memory. FCTs are issued when there is room in the FIFO, without consideration for the size of the host system buffer allocated for data transfer from the FIFO.

A. Packet Overflow

Receive buffers on link end points have high and low memory limits, whether that memory is statically or dynamically allocated, and whether a linked list is contiguous or scattered in physical memory. Receive buffer overflow is very damaging, so a high-availability system should seek to avoid that situation with a hardware mechanism of some sort. The host system may be removed from the receive front end by several layers complicating the connection between receive FIFO and receive buffer. The complexity of the chain may be inadequate to prevent receive buffer overflow or to avoid an intricate recovery.

B. GOES-R Spacecraft Receive Buffer Chain

The BAE SpaceWire ASIC is used in the GOES-R spacecraft to interface to the instruments, and Figure 2 illustrates the Application Specific Integrated Circuit (ASIC) cores involved in transferring incoming telemetry to system Static Random Access Memory (SRAM).

The Rx FIFO is connected to a Receive Interface (RIF), which controls a Direct Memory Access (DMA) engine to transfer data through the On-Chip Bus (OCB), to a Memory Controller (MEMCTL) which ultimately writes the packet into system SRAM.

Working from left to right in the Figure 2 example, there are 5 equally sized memory regions, MR1 through MR5, in SRAM. Each region has been sized to accommodate the maximum packet expected to be received. In this example, the memory is contiguous for two of the regions but is otherwise scattered. Incoming packets will be written first to MR1, then MR2 and so on, with MR5 linking back to MR1.

In typical producer-consumer fashion, each region would not be overwritten until consumed by the host system. Simple linked-list buffers should not be employed if there's any possibility of overwriting a buffer until it has been completely consumed. A non-linked list of descriptors, albeit with host software intervention, can be fashioned into a scatter-gather controller.

On the BAE SpaceWire ASIC, a receive descriptor is constructed by the host software to point to the target buffer in SRAM, by address and by length. The address of that descriptor is written to a RIF register and the RIF starts the process, which terminates when either the specified buffer length is reached or an end-of-packet marker is received. This

effectively prevents the designated receive buffer from overflow but does not terminate or spill the packet. Remaining packet data will consume additional buffers until an end of packet marker is received, complicating recovery.

C. Programmable Per-Port Maximum Packet Length

On GOES-R, the GSFC-designed router core embedded within the BAE SpaceWire ASIC includes an additional feature to prevent overflow on receive, and to prevent a stall due to blabbering transmit; a maximum packet length feature. This feature, if enabled, will disconnect the link and append an Error End of Packet (EEP) to any packet that exceeds the programmable maximum length. This will of course also spill the remainder of the errant packet. With the BAE SpaceWire ASIC, the limit applies to all ports, but ideally each port would have separate limits. The other router used on GOES-R, the Aeroflex 4-port router, has no such feature.

III. Receive Buffer Depth & Router Defect

Receive FIFO depth of a router may be configurable within a soft core for an FPGA. During development, pipeline side-effects should be taken into consideration to avoid potential stalls and data dropout. Router defects may exist which may cause a stall which will only clear when the receive buffer depth is adequate to compensate for the defect.

A. Logic Value vs. On-The-Wire Value

The value of a transmitter's credit count may be different in the logic of a transmit block then the value on the wire due to pipeline delays, synchronization delays and logic delays. A receive FIFO configured to a depth of only 16 bytes, and with a one byte pipeline delay, may initially transmit 2 FCT's, but not issue a successive FCT until 9 bytes are transmitted to it, and remain one byte delayed thereafter.

The transmitter may also have logic delays that cause its internal credit count to fall behind the value on-the-wire.

B. Router Transmit Block Defect

The Aeroflex 4-port router designed into GOES-R had a latent defect that was not exposed until integration testing with an instrument that had configured a receive FIFO depth of only 16 bytes. When the router's internal credit count transitioned to zero on the same cycle that an FCT was received, the router would stall due to the defect. The router would resume transmission when another FCT was received.

C. Indefinite Stall Condition

Although the router could break the stall when yet another FCT was received, the instrument configuration prevented further FCT transmit due to the shallow receive buffer depth, combined with the receive pipeline delay, causing the stall to last indefinitely.

D. GOES-R Stall Resolution

To avoid a reconfiguration of the instrument's FPGA, the transmit speed through the router was slowed to avoid the stall condition, which occurred only when the internal credit count transitioned to zero on the same clock as when an FCT was received. By slowing the transmit clock, the router's internal logic no longer lagged behind the on-the-wire value. Subsequent builds of the instrument did incorporate a deeper buffer to further mitigate the problem.

IV. PIPELINING PITFALLS

Pipeline stages were in part responsible for the problem described above, and is the main culprit in another issue encountered with GOES-R.

To avoid buffer overrun a producer-consumer buffering model was employed by the GOES-R spacecraft. Unlike linked-list operation, there is a time gap (latency) from DMA completion until the RIF is programmed with the next receive buffer. The bug, described below, was never observed when in linked-list mode.

A. Problem Description

Under nominal telemetry conditions a receive buffer was made available via the RIF (see Figure 2 above) prior to filling the Rx FIFO. Telemetry bursts, however, would exhaust the supply of receive buffers in SRAM until the downlink (consumer) could catch up. On occasion, the Rx FIFO and a 4-byte pipleline stage (not shown) would fill before a newly-freed buffer could be assigned to accept the packet via the RIF. Data did not overflow from the Rx FIFO because credit count depletion would stall the packet. There was a bug, however, in the pipeline stage that could drop those leading 4 bytes from the incoming packet. The GRDDP CRC would match, but half of the GRDDP header would be missing from the receive buffer. Several methods were utilized to address the bug.

B. GRDDP Transmit Retry

The GRDDP retry mechanism, for normal data packets, assures that those cropped packets will be retried, since header checks fail and the packets would not be acknowledged. Network traffic is increased, however, and dropouts of urgent message packets are possible since they are not retried.

C. Buffer Utilization

There wasn't enough physical memory to allow linked-list operation, but all remaining SRAM was dedicated for receive buffers, which helped quite a bit, but was not enough. Another technique considered was to dynamically utilize the allocated receive buffer space vs. a ring of fixed-size buffers. With this method, the start address for the next packet would depend on the size of the current packet, rounded up per DMA constraints. Pending on downlink transfer completion could be reduced or eliminated given the increase in number of buffers. While a sound idea, it was more complex, and would lead to additional processing latency.

D. Processing Delay Reduction

The assignment of a buffer freed by the downlink to the RIF had been a function of the main processing loop. By moving that function to an ISR context the mechanism behaved more like a linked-list. The addition of this latency reduction proved a sufficient workaround.

CONCLUSION

Real-world systems may be vulnerable to serious faults that can result even when there is no apparent violation of the ECSS standard. Additional capabilities are required of routers and nodes to avoid these pitfalls. All components in a system must be thoroughly researched, including the experience gained with those components by others.

REFERENCES

[1] A. Krimchansky, W. Anderson, C. Bearer, "The Geostationary Operational Satellite R Series SpaceWire Based Data System Architecture", NASA Goddard Space Flight Center, 2010

[2] European Cooperation for Space Standardization, ECSS-E-50-12A, "SpaceWire – Links, Nodes, Routers and Networks", 2003

[3] NASA Goddard Space Flight Center GOES-R Project "GOES-R Reliable Data Delivery Protocol", 417‐R‐RTP‐0050 Version 2.1, 2008

[4] L. Haynes, G. Rakow, "BAE SYSTEMS SpaceWire Router Specification: 4 Port, 2 External Interface", 2005

[5] J. Marshall, S. Santee, M. Hanley, J. Robertson, D. Stanley, "Leveraging SpaceWire Network Prototyping to Create Flexible SpaceWire Components and Support Software", 2011

[6] Aeroflex, "UT200SpW4RTR 4-Port SpaceWire Router Preliminary Datasheet", 2013

New Approaches for DC Balanced SpaceWire

Session: SpaceWire Networks and Protocols, Short Paper

Alex Kisin
NASA Goddard Space Flight Center/ASRC/AS&D
Greenbelt, MD, USA
Alexander.B.Kisin@nasa.gov

Glenn Rakow
NASA Goddard Space Flight Center
Greenbelt, MD, USA
Glenn.P.Rakow@nasa.gov

***Abstract*— Direct Current (DC) line balanced SpaceWire is attractive for a number of reasons. Firstly, a DC line balanced interface provides the ability to isolate the physical layer with either a transformer or capacitor to achieve higher common mode voltage rejection and/or the complete galvanic isolation in the case of a transformer. Secondly, it provides the possibility to reduce the number of conductors and transceivers in the classical SpaceWire interface by half by eliminating the Strobe line. Depending on the modulator scheme – the clock data recovery frequency requirements may be only twice that of the transmit clock, or even match the transmit clock: depending on a Field Programmable Gate Array (FPGA) decoder design.**

In this paper, several different implementation scenarios will be discussed. Two of these scenarios are backward compatible with the existing SpaceWire hardware standards except for changes at the character level. Three other scenarios, while decreasing by half the standard SpaceWire hardware components, will require changes at both the character and signal levels and work with fixed rates. Other scenarios with variable data rates will require an additional SpaceWire interface handshake initialization sequence.

***Index Terms*— SpaceWire, DC balance, Line encoding, PRS**

I. Introduction

DC balanced data lines (also referred as AC coupled lines), where "0" and "1" ratio is 1 (or very close to 1) over a certain time stretch, allows data to go over capacitive or transformer barriers, thus creating better isolation for communication modules at different common ground potentials. Currently, these potentials difference depends on receiver common mode voltage rejection: for Low Voltage Differential Signaling (LVDS) receivers it is +5/–4V at best. Originally, the SpaceWire hardware protocol was designed for an easy clock extraction and was not designed with DC balance in mind [1]. Over the recent years there have been several attempts to create a DC balanced SpaceWire hardware protocol, but all of them either failed to create DC balanced Data and Strobe by an easy means [2], or rejected the Strobe line, thus forcing user to extract clock from Data using special FPGA based decoders [3].

Authors will try to review some new methods, both with and without the Strobe line.

II. Methods with Data and Strobe Lines

II.A. Dual Complementary Bytes

One of the simplest methods will be converting each Data byte in to 2 bytes, where 1st byte is itself, along with Data Control Flag (DCF) and Parity (P), while 2nd byte is 1st byte inversion, including DCF and Parity, as seen in Fig. 1 below:

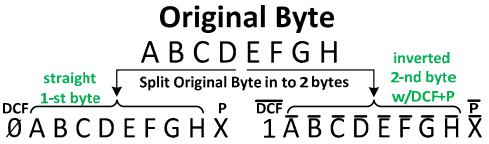


Fig. 1. Original Byte Split

From a first glance it is obvious that Data line will be balanced for the full 20-bit sequence, but will it be true too for a Strobe? Likely, it can be easily shown that for any number of complementary bits divisible by 4, a Strobe will also be balanced: because 01 or 10 clock sequences always place its "0" or "1" under the same complementary data positions of both bytes and the resulted Exclusive OR (XOR) Strobe bits will complement each other too: see Fig. 2 below with the encircled same color columns as an examples.

Simulation shows that both Data and Strobe lines will be balanced for any bits combinations. It is also important to note that the parity bit is not its classical SpaceWire implementation – it is a parity of the 9 previous bits including DCF and data byte, and it is irrelevant for this method whether parity bit is Even or Odd.

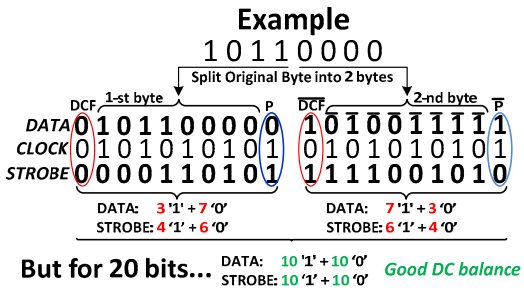


Fig. 2. Data and Strobe DC Balancing Example

Control characters and Time codes can be used the same old way, but will be also converted in to 2 complementary entities.

This method will also allow a simple error correction, where the user can select between 1st or 2nd received bytes: the one with a valid parity bit. A maximum stretch of the same bits sequence will be 18. If someone wants to reduce this stretch – they can try to play games of grouping bits and their complements between 2 bytes: like interleaving 2 adjacent bits

with their complements will shrink the maximum stretch to 6, but the error correction feature will be gone.

The major weakness of this scheme is its 100% overhead.

II.B. Pseudo Random Sequence (PRS) Modulation

Another method partially described in [2] is using PRS mixed with original data. It is easy to prove mathematically that every meaningful data stream mixed on a bit-by-bit basis with a random data stream becomes random itself. Furthermore, every further XOR operation with this newly minted random data will also produce random data. The PRS (organized on Linear Feedback Shift Registers, or LFSR [5]) is a close approximation for truly random data, and therefore, can be counted as such, especially for longer generated sequences. As a result, Data and Strobe created according to this might also be considered random and thus DC balanced.

Note: all LFSR sequences do not contain combination when all registers are equal to "0". This creates a slight misbalance in "0/1" ratio, because there can be a combination when all registers are "1". To solve this authors recommend to define an LFSR state when it is 1 clock away from being all "1" and then skip the all "1" state to the next consecutive state.

Control characters and Time codes everywhere, except the initial handshaking protocol, can be used "as is" because they are now part of PRS domain. However, during the handshake protocol, there is a possibility that a Null character, while being itself DC balanced and its Strobe image is not, will be transmitted by an Originator for a long time when Responder's receiver is not ready, will create the Strobe line bias drift and push the LVDS receiver input beyond its common mode voltage tolerances. To fix this problem – substitute the original Null character 01110100 with 10011100; as a result, Strobe will be changed from 00100001 to a DC balanced 11001001; FCT character will be changed to 1100 and its Strobe to a DC balanced 1001. As soon as the handshake phase is over – the system can revert back to its original Control characters.

And yes, there can be unique situations as it was noted in [2] when original Data or Clock may be mixed with matched or inversed PRS generator patterns, thus creating longer identical bits sequences. However, their probability is very low and their duration is short, plus any resulting bias drift of hardware lines can be mitigated by selecting LVDS receivers with wider common mode voltage input tolerances, such as Texas Instruments product: SN55LVDS33–SP [4].

Initialization handshake is shown on Fig. 3 below. There initialization Null and FCT characters should be selected by the previously described DC balance criteria; afterwards a user can revert back to using original control characters. It is worth to note that while being disabled during initialization – the LFSR's bit which is XOR mixed with data stream is preferred to be "0": for not to invert Null and FCT. The LFSR is enabled after 1st cargo "0" DCF is detected.

III. Methods with Data Line Only

Removing the Strobe line is potentially a good idea: it will increase wire bundle flexibility and reduce harness weight and complexity as well as on-board electronic hardware. But it also removes from the original SpaceWire its easy clock extraction, Nulls has always to fill gaps between data bursts to reduce bias drift, and instantaneous communication data rate switching has to be mitigated. However, using modern FPGA's will take care of these problems (see paper [3] References).

III.A. Dual Nibbles with 4b/6b Coding

Use 4b/6b [6] coding to substitute two of the original Data byte nibbles with two 6-bit symbols. Each symbol will contain an equal count of "0" and "1": 3. Number of permutations for 3 "1" bits in 6-bit symbol for 64 symbols group is 20, which means that each of 16 nibble's combinations will be assigned to its own DC balanced symbol, plus 4 extra symbols can be used as 4 SpaceWire original Control characters.

No DCF bit is needed: Data and Control characters symbols are now unique. And also no Parity bit: data integrity will be checked by 3 "1" per symbol, or 6 "1" per dual symbols.

This method will probably be the simplest Data only DC balancing scheme, with only 20% of data bandwidth overhead.

Strobe can't be used because it will not be DC balanced.

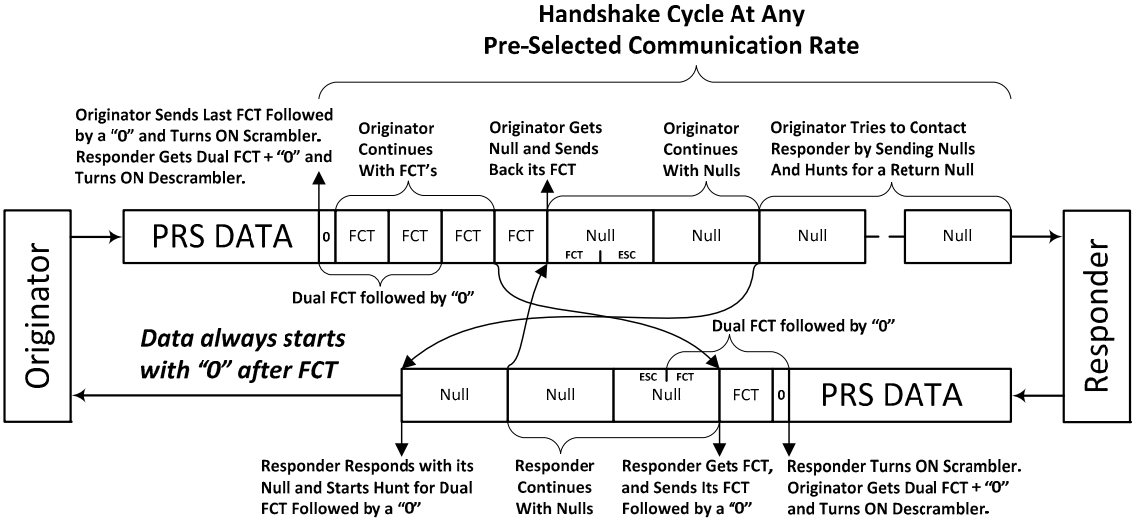


Fig. 3. Initialization Handshake Sequence with PRS

III.B. Fixed Rate with Dual Nibbles, Bytes or PRS

The fixed rate with Dual Bytes or PRS schemes are selected because they don't require clock frequency switching. Clock is extracted from an incoming Data stream using various FPGA techniques (see paper [3] References). Otherwise, these methods are basically the same as Data and Strobe PRS Modulation coding discussed in the above Section II.B.

III.C. Variable Rates with Dual Nibbles, Bytes or PRS

This method is also a derivative of the previous ones. Initial handshake at low rate will be done first and in a following cargo data Originator or Responder will notify each other about their desire to change data rate. After that Originator shall break the existing link, wait for at least 6.4us (during which time both sides adjust and stabilize their clock generators) and repeat their handshake at a new rate as shown in Fig. 4 below.

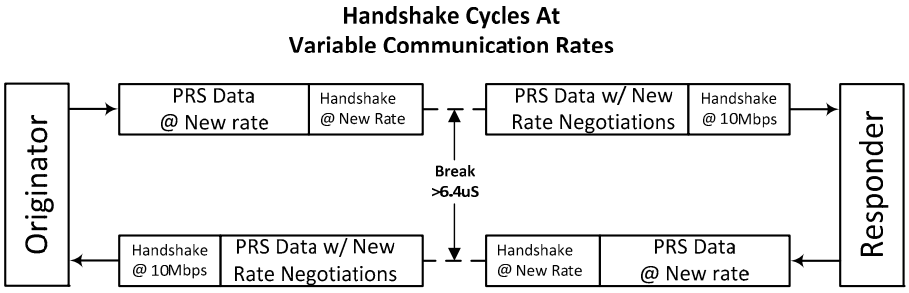


Fig. 4. Dual Handshake Sequence for Variable Rate PRS

IV. Summary

This paper presented an incremental design approach option to improve SpaceWire, yet leverages most of the existing FPGA based SpaceWire designs for moderate data rate applications that require or may benefit from electrical isolation. It also describes an additional way to further reduce the mass and flexibility of the SpaceWire cables for applications that are tight on space. Additionally, it provides a means to specify a common physical layer and one which could work with any protocol that uses DC balanced line codes.

Table I below shows what are in author's opinion brief characteristics of the above methods and also some not covered additional ones. They might be a little biased, but nevertheless will provide a design engineer with possible guidelines.

Table I. Methods Comparison Chart

Method	Description	Lines	RX Clock	Data Rate	Overhead vs. SpW	DC Balance	DC Quality	FPGA Implementation
1	Standard SpaceWire	Data and Strobe (D&S)	XOR-ed from D&S	Variable: as in original SpaceWire	0%	No	Terrible	Existing
2	Dual bytes (DB) encoding				100%	Yes	Very good	Easy
3	8b/20b [2]				100%	Yes	Good	Moderate-Complex
4	16b/30b [2]				50%	Yes	Good	Moderate-Complex
5	2 lines PRS modulation				0%	Yes	Good	Moderate
6	Fixed DB or PRS modulation	Data only	4-phase sampling, or others	Fixed: rate change requires dual handshaking	Same as in above D&S	Yes	Good	Moderate
7	Variable DB or PRS modulation					Yes	Good	Moderate
8	8b/10b or dual 4b/5b [7]				0%	Yes	Very good	Moderate-Complex
9	Dual nibble 4b/6b [6]				20%	Yes	Very good	Easy-Moderate
10	Manchester modulation				100%	Yes	Excellent	Easy-Moderate

References

[1] M. Suess, J. Ilstad, and W. Gasti: "Galvanic Isolation of SpaceWire Links: Requirements, Design Options and Limitations", 13th SpaceWire Working Meeting, 14-15 September 2009

[2] M. Epperly, S. Torno: "Galvanically Isolated SpaceWire", Proceedings of International SpaceWire Conference, Athens September 2014

[3] G. Rakow, A. Kisin: "Manchester Coding Option for SpaceWire", Proceedings of International SpaceWire Conference, Athens September 2014

[4] Texas Instruments: http://www.ti.com/product/sn55lvds33-sp

[5] Wikipedia: https://en.wikipedia.org/wiki/Linear-feedback_shift_register

[6] http://www.google.com/patents/EP0629068A1?cl=en

[7] https://en.wikipedia.org/wiki/4B5B

Components (Short)

Programmable SpaceWire interface with atom switch

Components, Short Paper

Hiroki Hihara[1,4], Nobuo Tamagawa[1],
Takayuki Imamura[1], Hisashi Sugaya[1]
[1]Space Engineering Division
NEC Space Technologies, Ltd.
Tokyo, Japan
{h-hihara@bc, n-tamagawa@pb,
t-imamura@pi, h-sugaya@bu}.jp.nec.com

Tadahiko Sugibayashi[2], Makoto Miyamura[2],
Toshitsugu Sakamoto[2], Munehiro Tada[2],
Hiromitsu Hada[2]
[2]Central Research Laboratories
NEC Corporation
Tsukuba, Japan
{t-sugibayashi@da, m-miyamura@aj, t-sakamoto@dp,
m-tada@bl, h-hada@bp}.jp.nec.com

Kazutoshi Wakabayashi[3]
[3]Services and Technologies Division
NEC Corporation
Kawasaki, Japan
wakaba@bl.jp.nec.com

Akira Iwasaki[4]
[4]Research Center for Advanced Science and Technology
The University of Tokyo
Tokyo, Japan
aiwasaki@sal.rcast.u-tokyo.ac.jp

***Abstract*—We developed a rewritable field programmable gate array (FPGA) using NanoBridge® that exploits atom switch technology. NanoBridge® is newly developed copper wiring technology with dynamic connection capability. Programmable circuitry with SpaceWire interface is realized without configuration memory cells for storing circuit connection information. It prevents single event effect caused by radiation and provides remarkably low power consumption. The first demonstration of the NanoBridge® FPGA on orbit by JAXA's program is planned in 2018.**

***Index Terms*— SpaceWire, Field Programmable Gate Array (FPGA), Atom Switch, Non Volatile, Rewritable, High Level Synthesis, Dynamically Reconfigurable Architecture**

I. INTRODUCTION

SpaceWire standard realizes compact and well organized interfaces for machine to machine (M2M) onboard information transmission. Since the hardware implementation of SpaceWire provides sufficient capability for intra-onboard communication of satellites, SpaceWire interface and application circuitry can be integrated into one field programmable gate array (FPGA). We developed a rewritable FPGA with NanoBridge®. It is newly developed copper wiring technology with dynamic connection capability using atom switches. The atom switch provides programmable capability without configuration memory cells for storing circuit connection information, and it can incorporate SpaceWire interface.

The connection of the atom switch can be controlled by practical voltage swing such as +/- 3.3V. The primitives of circuitry are connected with cupper ions through solid-electrolyte between ruthenium and cupper electrodes of the atom switches. 1,000 times of connection/disconnection are verified through thermal cycling test between -65 degree C and +150 degree C, which show sufficient programmability for onboard applications.

Conventional non-rewritable FPGA as well as application-specific integrated circuit (ASIC) is used for the hardware implementation of SpaceWire interface in order to provide low power consumption and reasonable radiation tolerance. The FPGA using NanoBridge® can provide equivalent low power consumption and radiation tolerance as non-rewritable FPGA and ASIC.

Some FPGAs are rewritable, whereas their power consumption and radiation tolerance are increased, because circuit connection information is stored in configuration memories like electrically erasable programmable read-only memories (EEPROMs) or static random access memories (SRAMs). Single event upset (SEU) on the memories is a major concern especially for SRAM. Thus memory patrol and/or memory scrubbing are mandatory, that requires additional resources outside those FPGAs. Atom switches used in NanoBridge® FPGA are SEU free, because they do not have configuration memories while maintaining programming capability of circuitry connection information. Irradiation tests of NanoBridge® FPGA have been done with some radiation sources without showing any SEUs in wiring layers. Therefore additional resource is not required in order to keep radiation tolerance.

The first demonstration of the NanoBridge® FPGA on orbit by JAXA's program is planned in 2018.

II. NANOBRIDGE®

The basic configuration of NanoBridge® FPGA is composed of complementary atom switches (CAS) [1, 2, 3, 4] as shown in Fig. 1 (a). The CAS consists of two copper atom switches with the polymer solid electrolyte (PSE) sandwiched

between two metals (Cu and Ru) as shown in Fig. 1 (b). The switch turns on or off when a nanometer-scale metallic bridge either appears or disappears inside a PSE film by biasing voltages. When a positive voltage like +3.3V is applied to a Cu electrode described in Fig. 1 (b), the Cu+ ions are supplied from the Cu electrode to PSE. The Cu+ ions are neutralized and precipitated at the Ru electrode. Subsequently, the precipitated Cu forms conducting bridges between the two electrodes, thus changing the conductance to an ON state. Conversely, by applying a negative voltage like -3.3V to the Cu electrode, the Cu bridge is ionized and disappears, resulting in an OFF state. On and off state are kept during an operational power supply voltage from around 1V to 2V is applied. Each state is nonvolatile and the switching between the two states is repeatable.

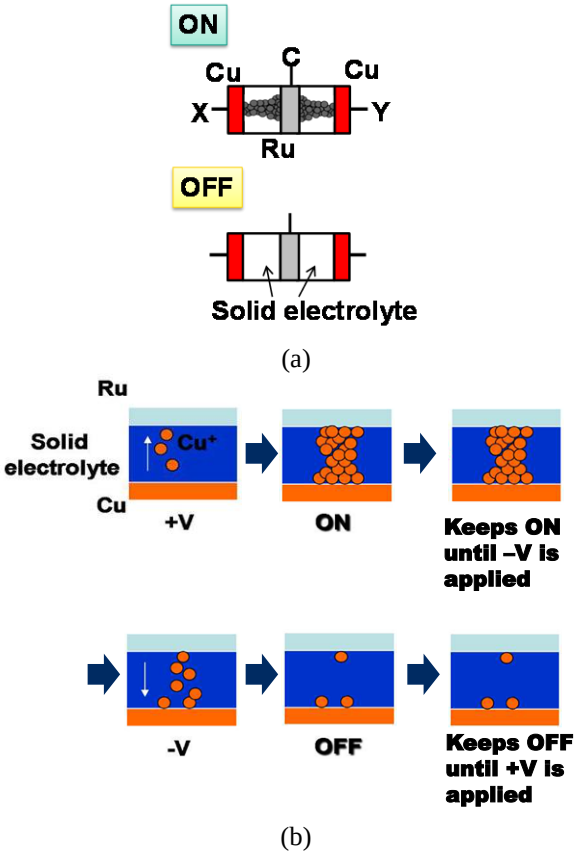


Fig. 1. Switching behavior of NanoBridge®

A Cu bridge in an atom switch can hardly be seen as shown in Fig. 2. Therefore the reverse engineering of NanoBridge® FPGA is difficult, and high level tamper-resistance is provided.

The atom switch layer is formed on an ordinary CMOS substrate with fabricated circuitries and wiring layers as shown in Fig. 3. There is no restriction for selecting CMOS substrates for NanoBridge® FPGA, therefore any kinds of CMOS substrates are applicable. Large scale integration (LSI) using NanoBridge® is fabricated with high reliability configuration. A series of switch of NanoBridge® are implemented in lookup tables in a FPGA, and thus high off state impedance is maintained.

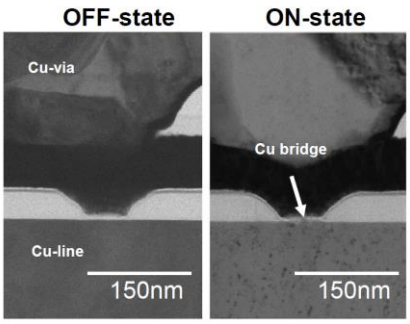
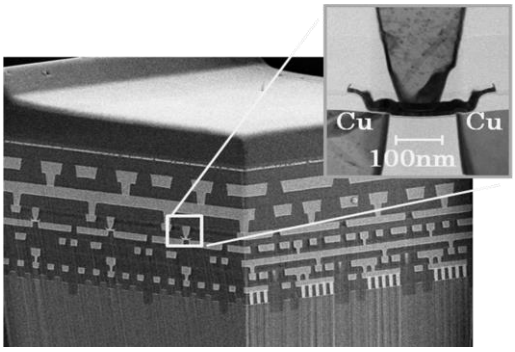
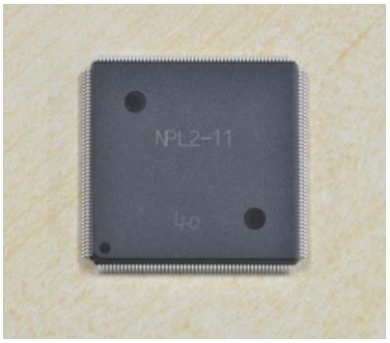


Fig. 2. Cu bridge formulation in NanoBridge®



(a)



(b)

Fig. 3. Programmable FPGA with NanoBridge®, (a) Cross-sectional FIB-SIM/TEM image, (b) Evaluated device with a plastic package.

III. SEE MITIGATION DESIGN FRAMEWORK

SEE mitigation is a major concern for satellite onboard equipment. Two types of memories are used in onboard components. One type is a program memory used in the central processing unit (CPU), and the other type is a configuration memory when a re-writable FPGA is used. We have to provide SEE mitigation design framework for both types of memories.

As for the latter case, NanoBridge® FPGA provides SEE free characteristics while maintaining re-writable functions. It provides programmable capability for circuitries without a configuration memory, therefore SEE originated from volatile memories are eliminated.

We use behavioral synthesize technology for providing sufficient radiation hardness against the former case using NanoBridge® FPGA. Application programs written in high level programming language like ANSI-C language are synthesized into register transfer level (RTL) like VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language) or Verilog source code by using a high level synthesis tool as CyberWorkBench (CWB) [5, 6, 7]. CWB is a well matured high level synthesis tool using behavioral synthesis technology. When we make application programs using CWB, the overhead in layout size is only a few percent larger than the design written in direct hand coded RTL. The application program is implemented on NanoBridge® FPGA directly. Program memories are eliminated, because CPU is no longer necessary for interface module using SpaceWire.

The two types of large volume memories as program memories for CPUs and configuration memories for re-writable FPGAs are eliminated by using CWB and NanoBridge® FPGA, and SEE mitigation is realized as shown in Fig. 4.

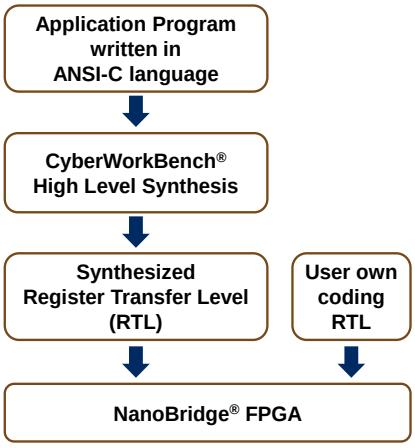


Fig. 4. SEE mitigation framework

We established a new design framework for applying CWB and NanoBridge® FPGA for system level SEE mitigation. The requirement of the reliability design flow is shown in Table 1. The design scheme is quoted from the dependable architecture described in [8, 9]. The robust fabric with traditional design method is incorporated in I/O primitives in the bottom layer for high reliability. Specifically, radiation hardened primitives like flip-flops are considered and implemented at circuit design level or semiconductor fabrication process design level in this layer. The radiation hardened primitives can be exploited as primitives for high reliability logic design.

The robust fabric design scheme is also useful for high reliability design of the fine grained layer. High reliability circuit implementations such as Triple Modular Redundancy (TMR), etc. are adopted on susceptible function blocks. If such kind of redundancy is adopted all over the chip, the high implementation density cannot be expected. The selective redundancy scheme described in [8, 9] is adopted in the design process, and the excessive resource overhead of power consumption, layout area, processing speed caused by redundancy is avoided.

As for course grained layer, a redundancy scheme for each function block is selected from the system FDIR design point of view. The use of TMR and/or CRAFTSYSTEM described in [10, 11] can be selected for gaining high reliability effectively in accordance with the reliability estimation results for a certain system FDIR design.

Proven communication network protocol is applicable on the topmost layer, which is the switch layer, in order to realize high reliability. The routing mechanism specified in the SpaceWire standard [12] is applied for implementing high reliability, because many off-the-shelf devices for space applications are available. It is another advantage that the system design can be performed in the scope of the open international standard.

TABLE I. THE REQUIREMENT SPECIFICATION OF RELIABILITY DESIGN SCHEME

Layer	Implementation	Remarks
Switch	Routing by SpaceWire regulations [12]	Implementation by system FDIR design
Coarse grained	Comparison decision using TMR [8, 9], and/or CRAFTSYSTEM [10, 11], etc.	Implementation by system FDIR design
Fine grained	Triple Modular redundancy with a voter (TMR), etc.	Implementation by system FDIR design, and the framework of robust fabric [8, 9]
I/O (Random Logic)	Radiation hardened libraries, etc.	The framework of robust fabric [8, 9]

We also established a design flow for the framework. The following six steps compose a design flow for mitigating SEE.

[Step 1]: Describe a system in high-level programming language as C language. The language should be used for application software development as well.

[Step 2]: Define coarse grained function blocks. The function blocks can be implemented as optimized hard macros using ASIC design tools. Typical dedicated functions as image compression, image recognition, and signal processing like FFT are implemented in addition to basic arithmetic operations. It is required on behavioral synthesis tools that the operations provided by coarse grained function blocks are used as mnemonics in program source codes. CWB realized this feature as functionalization.

[Step 3]: Generate source codes written in hardware description language (HDL) through behavioral synthesis. The hard macros defined in the step 2 should be exploited for

generating circuitry. This requirement specification is applied for the optimization of CWB during our study.

[Step 4]: This is a logic synthesis step, and this step is the same as conventional LSI and FPGA logic synthesis design process.

[Step 5]: This is a layout design step, and this is the same as conventional LSI and FPGA layout design process.

[Step 6]: The verification and validation step include delay analysis of layout design and back annotation based on the result of delay analysis.

Iterations are considered over step 5 and step 6. The back annotation going back to step 3 is also anticipated. Since the back annotation should be able to be handled in high level language, step 1 is also considered in back annotations.

In order to evaluate the design framework, we implemented signal processing functions of the infrared image sensor of AKATSUKI [13] onto one NanoBridge® FPGA. The signal processor of AKATSUKI consists of one CPU board with JAXA authorized 64bit microprocessor HR5000 and one FPGA board with two ACTEL RTSX72SU FPGAs. The dedicated operations as superposition, mean, and median with normalized functions are successfully implemented in one chip using the design framework. The nominal power consumption of the original signal processor is estimated as 4.9W. The measured power consumption of our one-chip processor element implementation on a NanoBridge® FPGA chip is 18mW. The reduction in circuitry also comes from the fact that whole processor functions are implemented in one chip, and interface circuits over devices are eliminated.

IV. Conclusion

Programmable SpaceWire interface with atom switch is realized using NanoBridge® technology. High level programming language usability is also maintained with newly established design framework. Single event effect of memories caused by onboard environment in orbit is avoided, because program memories in CPU and configuration memories for re-writable FPGAs are eliminated. Significant power reduction is also realized, which enables embedding a device computing function into each sensor detector.

Acknowledgment

This work was supported by the Grant-in-Aid for Energy and Environmental New Technology Leading Program of the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

Makimoto's wave [14] is considered as a reference, because the technology trend shown by Makimoto's wave is expected to be promising for next generation processor architecture. Authors thank Dr. Makimoto for his precious advice, and we continue the evaluation of our design framework aiming at being one candidate of Highly Flexible Super Integration (HFSI) shown in the technology trend in the wave.

References

[1] T. Sakamoto, S. Kaeriyama, M. Mizuno, K. Terabe, T. Hasegawa, and M. Aono, "NanoBridge Technology for Reconfigurable LSI," NEC Tech. J. vol. 2(1), pp. 72-75, 2007.

[2] M. Tada, K. Okamoto, T. Sakamoto, M. Miyamura, N. Banno, and H. Hada, "Polymer Solid-Electrolyte (PSE) Switch Embedded on CMOS for Nonvolatile Crossbar Switch," IEEE Transactions on Electron Devices, vol. 58, 12, pp. 4398-4405, 2011.

[3] M. Miyamura, M. Tada, T. Sakamoto, N. Banno, K. Okamoto, N. Iguchi, and H. Hada, "First Demonstration of Low-power Nonvolatile Programmable Logic Using Complementary Atom Switch compatible for Fully Automated Design Flow" IEEE International Electron Devices Meeting, pp. 247-250, 2012.

[4] M. Tada, T. Sakamoto, M. Miyamura, N. Banno, K. Okamoto, N. Iguchi, and H. Hada, "Improved OFF-State Reliability of Nonvolatile Resistive Switch With Low Programming Voltage," IEEE Trans. on Electron Devices, vol. 59, pp. 2357-2362, 2012.

[5] K. Wakabayashi, and C. Schafer, B, "All-in-C" Behavioral Synthesis and Verification with CyberWokBench., ed. P. Coussy and A. Morawiec, XVI, ISBN: 978-1-4020-8587-1, Chapter 7, Springer, pp. 113, 2008.

[6] K. Wakabayashi, and T. Okamoto, "C-based SoC Design Flow and EDA Tools," IEEE Tran. On CAD, pp. 1507-1522, 2000.

[7] K. Wakabayashi, "CyberWorkBench: Integrated design environment based on C-based behavior synthesis and verification," IEEE VLSI-TSA, pp. 173-176, 2005.

[8] D. Alnajjar, H. Konoura, Y. Mitsuyama, H. Shimada, K. Kobayashi, H. Kanbara, H. Ochi, T. Imagawa, S. Noda, K. Wakabayashi, M. Hashimoto, T. Onoye, and H. Onodera, "Reliability-Configurable Mixed-Grained Reconfigurable Array Supporting C-To-Array Mapping and Its Radiation Testing," Proc. IEEE A-SSCC, pp. 313-316, 2013.

[9] H. Konoura, D. Alnajjar, Y. Mitsuyama, H. Shimada, K. Kobayashi, H. Kanbara, H. Ochi, T. Imagawa, K. Wakabayashi, M. Hashimoto, T. Onoye, and H. Onodera, "Reliability-Configurable Mixed-Grained Reconfigurable Array Supporting C-based Design and Its Irradiation Testing," IEICE Trans. Fundamentals, vo. E97.A(12), pp. 2518, 2014.

[10] H. Hihara, M. Ohtsuka, Y. Mizushima, T. Morisato, T. Ohshima, H. Miyoshi, and K. Baba, "Space-born Fault tolerant Computers." Information Processing, vol. 35, No. 6, pp. 497-503, June 1994.

[11] H. Hihara, K. Yamada, M. Adachi, K. Mitani, M. Akiyama, and K. Hama, "Autonomous Fault Tolerant Computer for SERVIS-2 Satellite.", Technical Report of IEICE, DC2002-75, vol. 102, no. 492, pp. 19-24, December 2002.

[12] ECSS-E-ST-50-12C, "SpaceWire – Links, nodes, routers and networks," ECSS Secretariat, ESAESTEC, Requirements & Standards Division, 2008.

[13] T. Fukuhara, M. Taguchi, T. Imamura, M. Nakamura, M. Ueno, M. Suzuki, N. Iwagami, M. Sato, K. Mitsuyama, G. Hashimoto, L., R. Ohshima, T. Kouyama, H. Ando, and M. Futaguchi, "LIR: Longwave Infrared Camera onboard the Venus orbiter Akatsuki," Earth Planets Space, vol. 63, pp. 1009-1010, 2011.

[14] T. Makimoto, "Implications of Makimoto's Wave," IEEE Computer, vol. 46(12), pp. 32-37, 2013.

GR740 SpaceWire Router Validation Methodology and Results

SpaceWire Components, Long Paper

Magnus Hjorth, Javier Jalle, Felix Siegle, Jan Andersson, Sandi Habinc
Cobham Gaisler AB
Gothenburg, Sweden
[magnus | javier | felix | jan | sandi] @gaisler.com

Roland Weigand
European Space Agency
Noordwijk, The Netherlands
roland.weigand@esa.int

***Abstract*—The GR740 is a quad-core space-grade processor that includes a SpaceWire router with eight external ports. The validation results are presented to show effects of the integration of a SpaceWire router in a microprocessor system and the validation methodology is described to show one way of characterize timing performances SpaceWire links during production tests.**

***Index Terms*—SpaceWire, router, space-grade processor, timing characterization, production test**

I. Introduction

The GR740 is a multi-core microprocessor ASIC device based on the LEON4FT processor core [1] and the European Next Generation Microprocessor (NGMP) architecture [2], designed and developed by Cobham Gaisler. The GR740 ASIC is implemented on STMicroelectronics' C65SPACE technology platform [3] and is available in a hermetically sealed LGA625 package. The development of the GR740 is the outcome of the European Space Agency's initiative to develop the NGMP, and delivers a significant performance increase compared to earlier generations of European space processors.

Engineering models have been manufactured in 2015 and functionally validated during 2016.

The GR740 integrates a SpaceWire router with four internal ports, which are connected to the on-chip AMBA bus system, and eight external ports with on-chip LVDS.

The validation effort for the GR740 covers several aspects of the SpaceWire router implementation. User cases and performance measurements demonstrate capabilities of interfaces to the on-chip microprocessor system to create a high-throughput inter-processor link. The timing characterization methodology shows how testing of timing parameters can be performed as part of production tests.

II. Background

The LEON project was started by the European Space Agency in late 1997 to study and develop a high-performance processor to be used in European space projects. The objectives for the project were to provide an open, portable and non-proprietary processor design, capable to meet future requirements for performance, software compatibility and low system cost. Another objective was to be able to manufacture in a Single Event Upset (SEU) sensitive semiconductor process. To maintain correct operation in the presence of SEUs, extensive error detection and error handling functions were needed. The goals have been to detect and tolerate one error in any register without software intervention, and to suppress effects from Single Event Transient (SET) errors in combinational logic.

The LEON IP-core family includes the first LEON1 VHSIC Hardware Description Language (VHDL) design that was used in the LEONExpress test chip developed in 0.35 µm technology to prove the fault tolerance concept. The second LEON2 VHDL design was used in the processor device AT697 from Atmel (F) and various system-on-chip devices. These two LEON IP- core implementations were developed by ESA. Gaisler Research, now Cobham Gaisler, developed the third (LEON3) and fourth (LEON4) designs that are used in a number of avionics systems and also in the commercial sector.

Following the development of the TSC695 (ERC32) and AT697 processor components in 0.5 and 0.18 µm technology respectively, ESA has initiated the NGMP activity targeting a European Deep Sub-Micron (DSM) technology in order to meet increasing requirements on performance and to ensure the supply of European space processors. Cobham Gaisler, at the time Aeroflex Gaisler, was selected to develop the NGMP system that is centered around the new LEON4FT processor.

After extensive FPGA prototyping, a functional prototype was developed on commercial technology (eASIC Nextreme2)

for early SW development and user evaluation. Throughout 2014 / 2015, the design was ported to and manufactured in the C65SPACE platform from STMicroelectronics [3]. Besides the chip development, the existing SPARC software development environment has been extended for multi-core with updates to compiler, simulator, debug monitor and profiling tools, an SMP version of the RTEMS operating system and hypervisor development.

III. GR740 System Architecture

The four LEON4FT processors are connected to a shared bus, which connects to a 2 MiB Error Detection And Correction (EDAC) protected Level-2 cache before reaching external EDAC protected SDRAM. Each LEON4FT processor has a dedicated pipelined IEEE- 754 floating-point unit. The design makes use of extensive clock gating and the processors can be put into a sleep mode to conserve power when some or all processor cores are unused.

The main communication interfaces of the device include eight external SpaceWire ports connected to an on-chip SpaceWire router, two 10/100/1000 Mbit Ethernet ports, MIL-STD-1553B and 32-bit PCI. Two serial ports and two CAN-bus ports are also available.

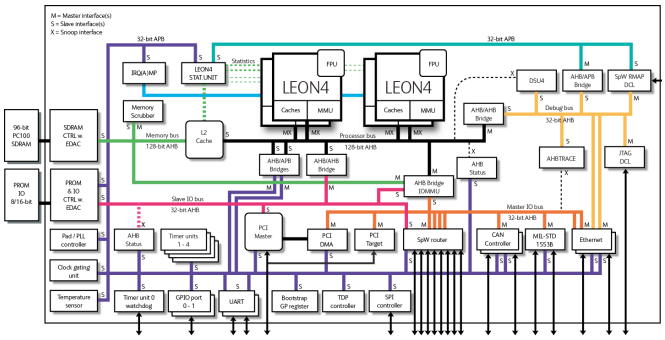


Fig. 1 GR740 Block diagram

The four parallel CPU / FPU cores, each running on dedicated separate instruction and data L1 caches (Harvard architecture), at 250 MHz clock frequency, can theoretically provide up to 1 Gflop/s in single or double precision.

The NGMP architecture has already been evaluated in a practical exercise where the GAIA Video Processing Unit (VPU) application, a real space payload application, was adapted to take advantage of a multi- core system and then benchmarked on an NGMP prototype system [4]. The conclusion from this exercise was that the GR740 would be fast enough to run the GAIA VPU application, likely at a significantly lower power budget than the flight computer that was used in the satellite.

The SpaceWire router in the GR740 has four internal ports, which are connected to the on-chip AMBA bus system, and eight external ports with on-chip LVDS. The external ports support cold-spare functionality.

The device can be configured via bootstrap signals so that the router is either clock gated off or enabled after power-on and reset. The Remote Memory Access Protocol (RMAP) targets are enabled if the router is enabled after reset, which means that SpaceWire can be used to remote boot the device by having an external host connect to the GR740 to upload software and then enable one or several processor cores. In case external RMAP traffic, or any type of direct memory access from the SpaceWire router, is unwanted then the systems IO memory management unit (IOMMU) can be used for address protection and address translation between the router's on-chip bus interfaces and the on-chip memory space.

The list below summarizes the key building blocks of the GR740 system architecture:

128-bit Processor AHB bus:

- 4x LEON4FT
  - 16 + 16 KiB write-through cache with LRU replacement.
  - SPARC Reference MMU. Physical snooping.
  - 32-bit MUL/DIV.
  - GRFPU floating-point unit
- 2 MiB Shared L2 write-back cache with memory access protection (fence registers), cache-way locking and partitioning.

128-bit Memory AHB bus:

- 1x 64-bit data SDRAM PC100 memory inter- face with Reed-Solomon ECC (with 16 or 32 check bits)
- 1x Memory scrubber

32-bit Master I/O AHB bus:

- SpaceWire router with eight external ports and four AMBA ports, with RMAP @ 300 Mbit/s
- 2x 10/100/1000 Mbit Ethernet interface with MII/GMII PHY interface
- 1x 32-bit PCI target interface @ 33 MHz
- MIL-STD-1553B interface

32-bit Slave I/O AHB buses:

- 1x 32-bit PCI master interface @ 33 MHz with DMA controller mapped to the Master I/O bus
- 1x 8/16-bit PROM/IO controller with BCH ECC
- 2x 32-bit AHB to APB bridge connecting:
- 5x General purpose timer unit
- 2x General purpose I/O port
- 2x 8-bit UART interface
- 1x Multiprocessor interrupt controller
- 2x AHB status register
- 1x Clock gating control unit
- 1x LEON4 statistical unit (perf. counters)
- 1x SPI master/slave controller
- PLL and pad control units
- 1x Temperature sensor
- SpaceWire Time Distribution Protocol controller

32-bit Debug AHB Bus

- 1x Debug support unit
- 1x JTAG debug link
- 1x SpaceWire RMAP target
- 1x AHB trace buffer, tracing Master I/O bus
- 1x PCI trace buffer:

IV. Radiation Tolerance Features

The GR740 is implemented on STMicroelectronics 65nm bulk CMOS process using the C65SPACE cell libraries developed and characterized by ST within the KIPSAT initiative [3]. The cell libraries contain standard cells (both radiation-hardened and standard variants), SRAM blocks, PLL and IO buffers. The cells used are specified for a temperature range of -40 to +125 degrees C and up to 20 years of aging effects.

Layout and back-end part of the design realization has been performed by ST using specific design rules and layout techniques developed for the C65SPACE platform.

All of the on-chip memories in the design, with the exception of memories used only for debugging, are used with single-bit error tolerance (of varying kind) to handle memory SEUs (Single Event Upsets). The memory blocks have been built with sufficiently high bit multiplexing factor to avoid multi-bit upsets on the same address due to a single event.

The level-1 cache memories inside the processor core use parity protection with transparent re-fetch from level-2 cache in case of detected error. Since the LEON4FT employs write-through caching, bad L1 cache lines can be invalidated and re-fetched without ever losing any written data. For the shared level-2 cache data memories, which employ a write-back policy and therefore may contain data not yet in memory, single-error correcting, double-error detecting (SECDED) protection using Bose-Chaudhuri- Hocquenghem (BCH) error correcting codes is used together with programmable periodic scrubbing to prevent build-up of multiple SEU:s over time. Other memories in the design, used for instance as buffers for data in transit to and from I/O interfaces, use error correction based on either duplication with parity or triplication and majority voting of the data on each memory address, depending on which one is more efficient.

The baseline approach chosen for implementing the register transfer level (RTL) logic inside the device has been to use radiation-hardened flip-flops and hardened clock tree elements but standard combinatorial logic cells. This is a trade-off between hardness level and functional performance. Other options were considered and a trial layout was made where TMR was used for the entire design, however this was abandoned due to performance and power impact.

For the CPU integer and floating-point register files built out of flip-flops, a different hardening approach based on triplication on block level, with bit-by-bit voting on the register read data outputs, has been implemented. This approach provides hardness at the same level as a raw flip-flop level TMR but without the same performance overhead. Inside each instance, standard flip-flops are used in order to save area and power. The layout has been checked in the implementation process to ensure that the flip-flops holding identical data in the three copies had adequate spacing between them, and fixed up where needed, to avoid sensitivity to potential uncorrectable multi-bit upsets. The register file is not automatically scrubbed so all registers need to be written at regular intervals, which normally gets done anyway in applications as part of task switching. For very simple applications where there is a risk of keeping register values for a long time, a periodic re-write of the registers (copying the register back to itself) could be, for example, done in the timer interrupt handler.

Three radiation-hardened PLLs are used for clock generation in the design [5]. Each PLL may be individually bypassed, and the lock status can be monitored directly through output pins. The GR740 can be configured to either trigger a reset of the processors, or to keep going, in the event of losing lock. A dedicated PLL watchdog function clocked on the input clock has also been added. This provides the ability to detect the unlikely scenario where a PLL would fail and stop delivering a system clock to the processors and the rest of the system.

A dedicated PLL reprogramming unit has been designed to allow re-programming the PLL configurations from the software boot-loader. A lock-down function is implemented that blocks further reprogramming by application software until a full system reset is done, to prevent accidental reprogramming after boot-up. The PLL configuration unit has been implemented with both hardened flip-flops and TMR to get maximum protection against upsets in the PLL configuration.

V. Production Test Approach

The production tests an industrial and complete test solution at wafer level and post-assembly. Each sample is tested at all supply level ranges within the specified temperature ranges.

The tests include

- DC tests: IO levels, leakages, consumption
- IDDQ tests
- Scan
- Memory BIST
- Transition fault
- PLL tests
- Cold spare tests
- AC parameters

Wafer probing of the GR740 device is performed by STMicroelectronics. The test equipment used is a Automatic Test Equipment (ATE) tester together with a probe card that has been custom designed for the GR740. The main purpose of wafer probing is to detect and separate faulty dies before packaging and also statistics collection. The SpaceWire router implementation is covered by wafer probing test patterns but no specific SpaceWire tests are performed during wafer probing. The functionality of the SpaceWire router is tested by means of test patterns generated according to industry standard flows. Timing performance of the external SpaceWire ports depend on the IOs and characteristics of the SpaceWire clock generated by an on-chip PLL.

The implemented production tests that monitor timing performances of the external SpaceWire ports are divided into two parts: SpaceWire interface TX skew test and SpaceWire interface RX skew test.

The SpaceWire interface TX skew test verifies timing skew between outputs transmit data (TXD[x]) and transmit strobe (TXS[x]) is with specification for all eight links x=1..8. To do

this, the test pattern start and stops the SpaceWire links. The time measurement capability of the ATE tester is used to calculate the time from a known quiet time while the link is off, until the first rise/fall transition on TXD and TXS corresponding to the SpaceWire handshake when the link is turned on. The times measured for the two pins are then subtracted, and offset with the known waveform pattern, to obtain the skew, The measurement is done individually for each link and for each rise/fall edge combination, each combination tested 10 times and taking the worst value (8*4*10 iterations total).

The SpaceWire interface RX skew test verifies the minimum RXD/RXS period that can be detected by the SpaceWire receiver. This is done by testing at what minimum period a SpaceWire packet can be successfully received from the ATE tester.

The difference between the measured limit and the ideal limit of two times the router's receive clock (2x due to DDR sampling of incoming signals) determines the effective receive skew. The receive data and strobe signals are DDR sampled with a clock generated by the on-chip SpaceWire PLL. In this test, the reference clock for the SpaceWire PLL is fed with a free-running clock from the ATE relative to the received SpaceWire data, which then means that the phase of the sampling clock used by the SpaceWire codec will be varying randomly relative to the receive data and strobe signals during the test.

In order to monitor the received data, the test equipment interfaces with the GR740 system through a JTAG debug link that, through the use of a JTAG/AMBA AHB bridge can perform memory accesses on the on-chip bus. This allows complete control of the SpaceWire router. The Level-2 cache is configured to be used as on-chip memory to store the received SpaceWire data where the full packet contents can then be verified from the outside world.

VI. Functional Validation Overview

The functional validation tests are run at room temperature in a lab environment. Tests are typically controlled via an external debug monitor (Cobham Gaisler's GRMON2 software) and for the SpaceWire router the tests utilize the on-chip processors as well as an external system with SpaceWire interfaces to generate and validate traffic.

The functional validation effort for the GR740 builds on previous validation efforts on FPGA prototypes of the NGMP architecture and validation of the NGMP functional prototype, the LEON4-N2X device [6]. The existing prototype SpaceWire router tests focused on exercising the router's AMBA ports. For the GR740 validation effort, router testing was expanded to include traffic on SpaceWire links. The router's redundant link capability, multicast capability and priority capability was also tested.

All tests described below were performed with the internal SpaceWire fabric running at 400 MHz and the AMBA system running att 200 MHz. All SpaceWire links were configured to operate at a bitrate of 200 Mbit/s. The development board used was the GR-CPCI-GR740 board [7]. It can be noted that the tests do not require the on-chip SpaceWire fabric to run at 400 MHz. The setting was kept as it is the default configuration attained without reconfiguring the design's clock generation circuitry.

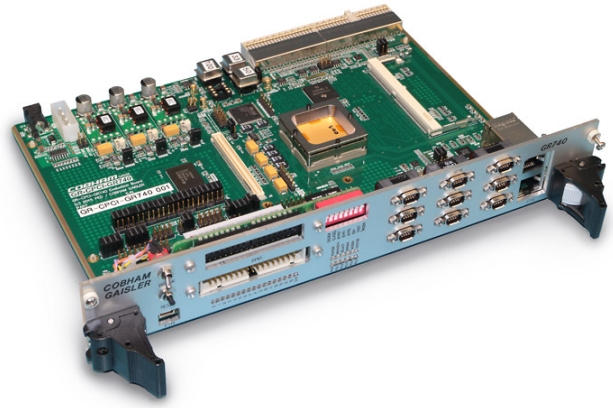


Fig. 2 GR-CPCI-GR740 Development board

VII. Functional Validation: All SpaceWire ports

To validate that all the SpaceWire ports of the SpaceWire router can handle both receive and transmit at a rate of 200 Mbit/s, each SpaceWire port was connected to another SpaceWire port. 4 MiB packets were then sent from an AMBA port, routed out onto a SpaceWire port, received at another SpaceWire port, and then routed to an AMBA port were the data was validated. This test was repeated so that all SpaceWire ports were utilized, and both path addresses and logical addresses were used for the packets.

VIII. Functional Validation: Group adaptive routing

The SpaceWire router supports group adaptive routing for all path addresses and logical addresses. Group adaptive routing means that packets can be routed through the network over different paths depending on which of the router's ports that are available when the packet arrives. For example, a packet with address 0x40 arrives at SpaceWire port 1 of the router, and address 0x40 is configured with group adaptive routing to SpaceWire port 2 and 3. The router will then route the packet to either port 2 or port 3 depending on which port becomes available first. If both ports are available, the router will send the packet on the port with the lowest port number.

The group adaptive routing mechanism was validated by connecting four SpaceWire ports together and then sending packets from an AMBA port where the address byte of the packets were configured with group adaptive routing to two of the four ports. When the packets arrived at the router again they were routed to another AMBA port. It was then verified that the packets arrived correctly as long as one of the two SpaceWire used as output ports were connected to another port. If none of the two SpaceWire ports used as output ports were connected then the packet was not received at the AMBA port used as destination. Group adaptive routing was also verified further in the packet distribution validation.

IX. Functional Validation: Packet distribution

Packet distribution - which means that data arriving at a input port is sent to multiple ports simultaneously - is supported by the SpaceWire router for both path addresses and logical addresses. This feature was validated by connecting four SpaceWire ports to each other and then sending a packet with two address bytes from an AMBA port. The first address byte was configured with header deletion and packet distribution out on the four SpaceWire ports, and the second address byte was configured with group adaptive routing to AMBA ports 0-3. When the packet was sent from the AMBA source port the first address byte was removed by the use of header deletion, and the packet was routed out onto the four SpaceWire ports. It was then verified that the four packets, arriving at one SpaceWire port each, was routed to one AMBA port each (because group adaptive routing was used for the second address byte). This test also adds additional validation of group adaptive routing since the test validates that group adaptive routing works when the destination ports are busy with transmitting data. The validation of group adaptive routing described above only validated the case when the destination links were not running.

X. Functional Validation: Priority routing

When packets are to be routed, each destination port is arbitrated individually using a two level priority. The priority is based on the first address byte of the incoming packet, and all path addresses and logical addresses can be assigned either a high or low priority. Round-robin is used when one or more packets with the same priority competes about the same destination port. The validation of the priority routing mechanism was done by enqueueing four different packets, each one from a different AMBA port, where all packets were to be routed out on the same SpaceWire port. Three of the packets contained an address that had been assigned a low priority, while the fourth packet contained an address with high priority. The SpaceWire port that the packets would be routed out onto was connected to another SpaceWire port of the router, and the second address byte in all packets was the path address of one of the AMBA ports (same for all packets so that the order could be observed). The three low priority packets were sent slightly before the high priority packet, and it was then validated at the destination AMBA port that the first packet received was the first low priority packet, followed by the high priority packet, and then followed by the two remaining low priority packets. It was also validated that if the high priority packet was instead changed to low priority it was received last of the four packets.

XI. Functional Validation: New features

The SpaceWire router implementation in the GR740 has several new features compared to previous prototype implementations of the NGMP architecture. The new features include support for 64 interrupt codes and time code propagation. Validation of these new features has been performed by reusing test developed for the GR718B 18x SpaceWire router device [8] and by reuse of existing tests available for the RTEMS operating system to test time code transmission.

XII. Functional Validation: User Case and Performance Validation

To demonstrate a user case and performance figures. An example was developed using an available Remote Memory Access Protocol (RMAP) stack and SpaceWire drivers for RTEMS. Using the example, which performance RMAP read and write accesses to a generic RMAP target. The data rates achieved between two GR740 systems with default configuration is around 20 Mbytes/s using one SpaceWire DMA channel (out of four) of one SpaceWire AMBA port (out of four) and one SpaceWire link (out of eight). During the test, one data array was transferred both ways (read and write) and the traffic was controlled by the LEON4FT in one of the systems.

The data rates achieved were measured at a high level, which means that they also consider the software overhead of performing the transmissions. This overhead can be further reduced optimizing the code for instance using a zero-copy driver. This was not done for two reasons; one to keep the example code to understand and to be able to provide the test case as an example for users of the device, and two to avoid a high optimization level that may not be practical to perform considering constraints of real world development efforts.

XIII. Radiation Tests of SpaceWire Router

The GR740 has undergone a radiation test campaign to verify its radiation performance against single event effects. Single-event testing with both heavy ions has been performed at the time of writing and tests with with protons is planned for to take place in the last quarter of 2016.

The single-event testing focuses both on raw upset rates of the various building blocks of the design and the error cross-section of the whole design in functional scenarios. The heavy-ion SEU test setup is based on the GR740 evaluation board [7] with the device wire-bonded to an unsealed ceramic LGA package, that is then mounted on the circuit board through a socket.

The ASIC has several test functions included to aid the radiation testing. The L1 and L2 caches have counters for corrected errors that can be monitored on the fly by test software. For the triplicated register files, flags have been added that are set when a voting mismatch has been detected to allow counting errors during test. For TID testing, a ring oscillator built out of standard cells is implemented on chip.

The test software is based on the “SEU32” test suite developed originally for LEON3FT. This software has been modified and expanded to handle the new architectural features of the GR740, such as the addition of level-2 cache. The software can be run from the flash memory of the evaluation board and communicates with a standard PC on the outside of the irradiation area through the GR740s serial ports.

The SEU32 was extended with a test specific for the on-chip SpaceWire router. The test consists of software that runs on one or several of the LEON4FT processors and sends

packets with an unique data over different ports that are assume to be connected to a loopback connector (or equivalent) so that the data that goes out comes in through the same port. The test checks that the transmission and the data arrived without errors. If errors are detected, a report is produced. Normal memory errors (status bits and counters for processor register file, Level-1 and Level-2 caches) are also monitored and reported.

The test initializes the SpaceWire router and its AMBA ports, creates and initializes transmitter and receiver descriptors for the packets and the data on it that contains a sequence of sequential numbers. The packets use path addressing that goes to each tested port and then back to the AMBA port. Since the loopback transmission takes some time, the CPU waits using a loop of NOP instructions that consumes always a fixed amount of time (adjusted to give time to the SpaceWire traffic) so that the test does not depend on timers or the SpaceWire cores themselves.

XIV. Conclusion

The GR740 validation effort has validated and characterized a SpaceWire router implemented as part of a space-grade microprocessor device. As part of the GR740 validation, production tests have been developed that allow characterization of the external SpaceWire links during production tests (wafer probing and post-packaging tests).

The GR740 device has been manufactured and is being validated in an activity funded and initiated by the European Space Agency. A technical note covering the functional validation results will be published online during the end of 2016. The results of the radiation test effort will also be published in a report once the radiation test campaign has been completed and the resulting report has been reviewed and approved by the agency.

Acknowledgment

The development and implementation of the GR740 device has been funded by the European Space Agency as part of the Next Generation Microprocessor activities.

References

[1] Cobham Gaisler. *GR740 Quad-Core LEON4FT SPARC V8 Processor* [Online]. Available: http://gaisler.com/gr740

[2] European Space Agency. *GR740: The ESA Next Generation Microprocessor (NGMP)* [Online]. Available: http://microelectronics.esa.int/ngmp

[3] P. Roche, G. Gasiot, S. Uznanski, J-M. Daveau, J. Torras-Flaquer, S. Clerc, and R. Harboe-Sørensen. *A Commercial 65 nm CMOS Technology for Space Applications: Heavy Ion, Proton and Gamma Test Results and Modeling*, IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 57, NO. 4, AUGUST 2010

[4] Cobham Gaisler. *RTEMS SMP Executive Summary: Development Environment for Future Leon Multi-core*" [Online] Available: http://microelectronics.esa.int/ngmp/RTEMS-SMP-ExecSummary-CGAislerASD-OAR.pdf

[5] F. Malou, G. Gasiot, R. Chevallier, L. Dugoujon, P. Roche: *TID and SEE Characterization of Rad- Hardened 1.2GHz PLL IP from New ST CMOS 65nm Space Technology*, IEEE Radiation Effects Data Workshop (REDW), July 2014

[6] Cobham Gaisler. *GR-CPCI-LEON4-N2X Quad-Core LEON4 Next Generation Microprocessor Evaluation Board* [Online]. Available: http://gaisler.com/gr-cpci-leon4-n2x

[7] Cobham Gaisler. *GR-CPCI-GR740 Quad-Core LEON4FT Development Board* [Online] Available: http://gaisler.com/gr740

[8] Cobham Gaisler. *GR718 Radiation-Tolerant 18x SpaceWire Router* [Online] Available: http://gaisler.com/gr718

High-Reliability SpaceWire Engine implemented on theSOISOC3 microprocessor

Components, Short Paper

Takanori Narita, Masahiro Taeda, Masahiro Kato, Masaki Kusano, Kazunori Masukawa
Applied Electronic Equipment Design Section,
Mitsubishi Heavy Industries Ltd (MHI),
1200, Higashi Tanaka, Komaki, Aichi, 485-8561, Japan
takanori_narita@mhi.co.jp, masahiro_taeda@mhi.co.jp
masahiro_kato@mhi.co.jp, masaki_kusano@mhi.co.jp,
kazunori_masukawa@mhi.co.jp

Takayuki Ishida, Seisuke Fukuda, Keiichi Matsuzaki, Tadayuki Takahashi
Institute of Space and Astronautical Science (ISAS),
Japan Aerospace Exploration Agency (JAXA)
3-1-1 Yoshinodai, Sagamihara Chuo-ku, Kanagawa, 252-5210, Japan
ishida.takayuki@jaxa.jp, fukuda@isas.jaxa.jp,
matsuzaki.keiichi@jaxa.jp, takahasi@astro.isas.jaxa.jp

Mitsutaka Takada, Hiroaki Takada,
Graduate School of Information Science, Nagoya University,
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan
mtakada@nces.is.nagoya-u.ac.jp, hiro@ertl.jp

Masaharu Nomachi
Research Center for Nuclear Physics, Osaka University,
1-1 Machikaneyamacho, Toyonaka, Osaka 560-0043, Japan
nomachi@rcnp.osaka-u.ac.jp

***Abstract*—SOISOC3 is our new space-grade system-on-chip processor which is currently being developed by MHI in partnership with JAXA. The chip is implemented on a 200 nm radiation hardened process based on the commercial SOI (Silicon On Insulator), so that can apply to the Space missions. This is the next generation system-on-chip processor upgraded from the currentSOISOC2 chip already used for ASTRO-H, ERG satellites, etc.**

SOISOC3 has a high-reliability SpaceWire engine, which MHI has developed, supporting for incoming SpaceWire standards for deterministic data delivery (SpaceWire-D), reliable data transfer service (SpaceWire-R), as well as high performance Remote Memory Access Protocol (RMAP) with Direct Memory Access (DMA) engine through a SpaceWire router. The SpaceWire engine is capable of acting as an RMAP initiator, target, or as a general purpose packet transmitter and receiver. Our developed SpaceWire-D and SpaceWire-R engine is mainly performed on hardware, and can achieve high accuracy scheduling and high performance, with less CPU load.

We'll introduce the outline and current status of our development with an prototype evaluation board exhibited at MHI booth.

***Index Terms*—SpaceWire, SpaceWire-D, SpaceWire-R, processor, system-on-chip, High-Reliability.**

I. Overview

Mitsubishi Heavy Industries (MHI) developed a space-grade system-on-chip processor, SOISOC2. SOISOC2 has a high performance CPU core and a basic SpaceWire engine which supports RMAP protocol. The chip is implemented on a 200 nm radiation hardened process based on the commercial Silicon On Insulator (SOI), so that can apply to the Space missions [1,2]. Fig.1 shows flight records of SOISOC2. SOISOC2 has been inside of a large number of satellites. SOISOC2 is also used in a commercial product, a radiation monitor as dual-use.

And then SOISOC3 is our new space-grade system-on-chip processor which is currently being developed by MHI in partnership with JAXA. This is the next generation system-on-chip processor upgraded from the current SOISOC2 chip.

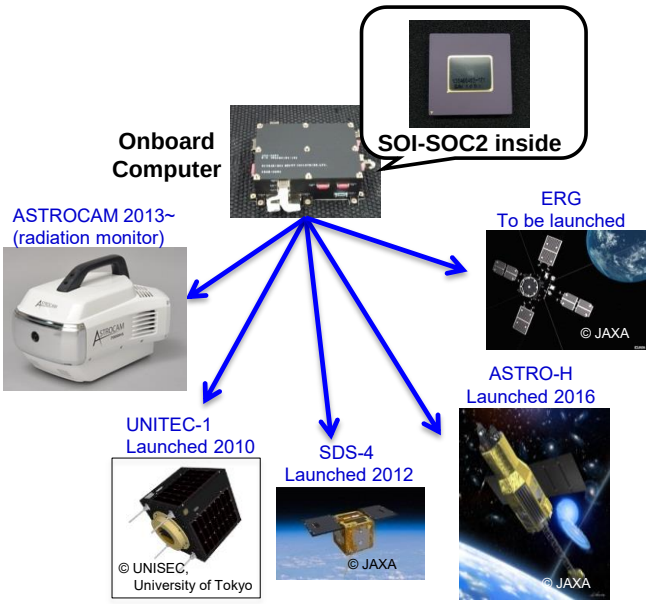


Fig. 1. Flight records of SOISOC2

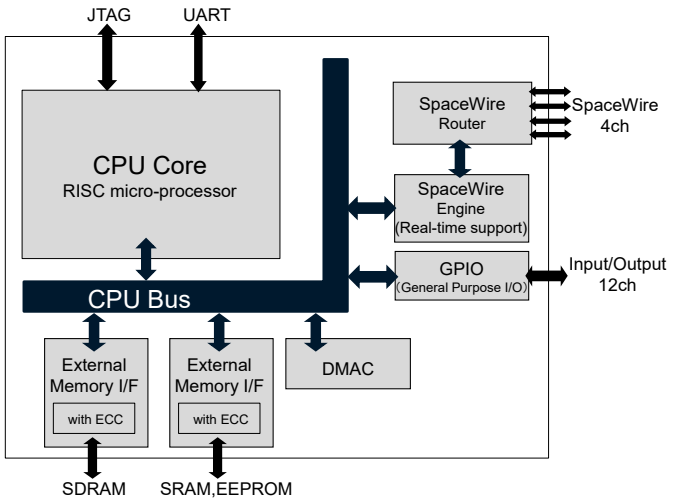


Fig. 2. Block diagram of SOISOC3

II. Specification

SOISOC3 maximizes the design property of SOISOC2 for steady development. TABLE I shows the specification of SOISOC3. Fig.2 shows the block diagram of SOISOC3.

SOISOC3 is same semiconductor process, CPU core, and memory interface as SOISOC2. The architecture of the CPU core is used in many high-reliability products. (e.g. Automotive products)

The SpaceWire engine is upgraded from SOISOC2. SOISOC3 has a high-reliability SpaceWire engine, which MHI has developed, supporting for incoming SpaceWire standards for deterministic data delivery (SpaceWire-D), reliable data transfer service (SpaceWire-R), as well as high performance Remote Memory Access Protocol (RMAP) with Direct Memory Access (DMA) engine [3,4,5].

TABLE I. Specification Of SOISOC3

Item		Specification	Remark
Semiconductor Processes		200nm SOI	Same Specification as SOISOC2
CPU Core		32bit RISC Processor Max. 100MIPS	Same Specification as SOISOC2
Memory Interface		SDRAM I/F and SRAM/EEPROM I/F	Same Specification as SOISOC2
SpaceWire engine		-	-
	Link Frequency	Max. 120MHz	Upgrade from SOISOC2 (~100MHz)
	External Port	4ch	Upgrade from SOISOC2 (3ch)
	Support Protocol	RMAP Raw SpaceWire-D SpaceWire-R	Upgrade from SOISOC2 (RMAP only)

III. Detail Of SpaceWire Engine

Our developed SpaceWire engine is mainly performed on hardware, and can achieve high accuracy scheduling and high performance, with less CPU load. Details of hardware support functions are below.

A. SpaceWire-D Hardware support

- RMAP Write

  When the SpaceWire engine receives a write command from a Initiator, the SpaceWire engine checks the header format (Key, Logical Address, Header CRC etc.) and writes the data to the internal register or external memory (SDRAM or SRAM or EEPROM) directly by using DMA engine with no CPU load.

  If requested in the write command, a write reply is sent by the SpaceWire engine back to the initiator of the write command or to some other node as defined by the header (Reply Address field).

- RMAP Read

  When the SpaceWire engine receives a read command from a Initiator, the SpaceWire engine checks the header format, reads the data from the internal register or external memory directly and sends a read reply to the initiator of the read command or to some other node.

- Timing Cotrol

  The SpaceWire engine compares time-slot numbers with send requests that is generated by SpaceWire-D Initiator (middleware) at each time-slot and controls timing of sending a read/write command to achieve high accuracy scheduling.

  Fig.3 shows one example, the flow chart of a transaction on a SpaceWire-D static bus service.

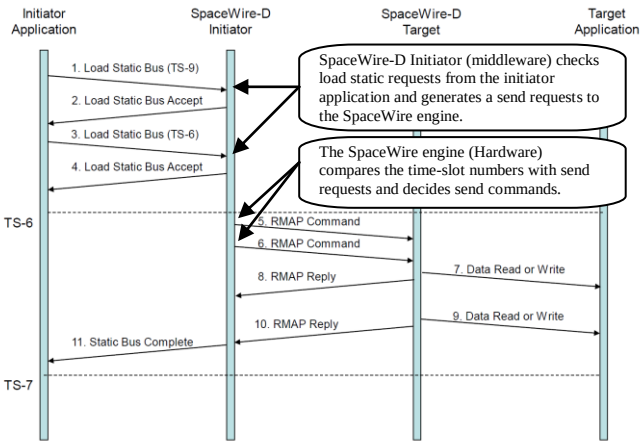


Fig. 3. Transaction on a static bus service (Ref. [1] pp. 71)

- Time-Code Watchdog

  The SpaceWire engine has hardware time-code watchdog timer. The SpaceWire engine checks for arrival of a time-code. In the event of an early or late time-code, the SpaceWire engine sets an error flag and an interrupt signal.

B. SpaceWire-R Hardware support

- Segmentation

  At a sending end, the SpaceWire engine breaks a service data unit (SDU) into smaller pieces by hardware so that each piece can be transmitted in a SpaceWire-R packet. At a receiving end, the SpaceWire engine reconstructs the original SDU from a series of received SpaceWire-R packets with no CPU load.

- Data Ack Reply

  Data ack packets are used to acknowledge receipt of Data. When the SpaceWire engine receives data packets, the SpaceWire engine checks data packets and replies data ack packets immediately.

IV. Software Interface

A real-time operating system called "TOPPERS HRP2 Kernel" and a board support package (BSP) which is included a SpaceWire middleware is available for SOISOC3 [6]. "TOPPERS" is based on the technical development result applied "ITRON". "TOPPERS" is used in many embedded devices.

Fig.4 shows protocol stack of SOISOC3. SOISOC3 middleware and hardware supports four types of protocols, RMAP, SpaceWire-D, SpaceWire-R, and Raw. The Raw protocol is consisted of the destination address field and the cargo field. A user application can set an original header and data as the cargo field. All protocols can select enable/disable of time slot scheduling. There also can be multiple plotocols on a same SpaceWire subnetwork by using time slot scheduling. (e.g. TS-6=SpW-R, TS-7=Raw, TS-8=SpW-D etc.)

SOISOC3 users can handle SpaceWire protocpls easily by using these software interface.

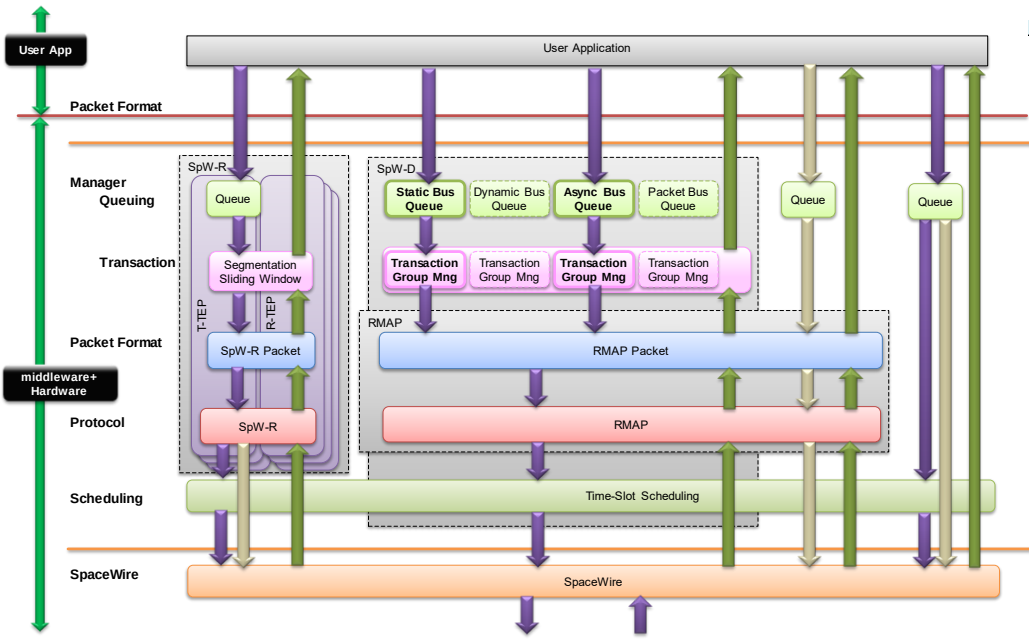


Fig. 4. Protocol stack of SOISOC3

V. Performance Of SpaceWire Engine

Fig.5 shows a picture of a evaluation board of SOISOC3. The evaluation board consists of FPGA and peripheral ICs. (LVDS transmitter, SDRAM, SRAM, EEPROM, regulator etc.) All hardware functions of SOISOC3 are included in FPGA.

For performance test of the SpaceWire Engine, one of the evaluation board is the initiator and the other is the target. A dummy application for performance test operates on middleware. A processing time of middleware and hardware is measured by SpW-link analyzer and logic analyzer.

TABLE II shows the test parameter. In the performance test of RMAP write/read, it issued 1000 RMAP write/read commands. Total data size is 1Mbyte. From the start of the first command transmission to the last command transmission, it takes 252 milliseconds, thus the result is 32.5 Mbps. (= 1Mbyte*8bit / 252 milliseconds)

In the performance test of SpaceWire-R, it issued 8 SDUs. Total data size is 8Mbyte. When segmentation size is 256byte, it takes 850 milliseconds, thus the result is 78.9 Mbps. In case of 512Byte, the result is 90.7 Mbps. In case of 1kByte, the result is 93.3 Mbps. For the SpaceWire-R hardware support, the result is good agreement with the value that had been obtained by theoretical calculation.

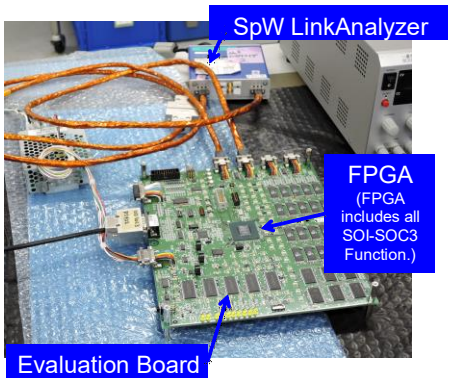


Fig. 5. Evaluation board of SOISOC3

TABLE II. Test Parameter

Parameter	Value	Remark
SpaceWire Link Rate	120MHz	Theoretical speed is 96Mbps.(=120*8/10)
Data Size of RMAP	1kByte*1000transaction	
Data Size of SpW-D	1Mbyte*8packets	
Segmentation Size of SpW-D	256/512/1kByte	

VI. Development Status

TABLE III shows the development status of SOISOC3. ASIC#1 design and manufacturing are ongoing. SOISOC3 is scheduled to be ready by the end of March 2018.

VII. Conclusion

We introduced the overview and development status of SOISOC3. SOISOC3 is the new ASIC for space products which has the high-reliability and the high performance SpaceWire engine. We consider that the chip will provide an efficient and cost-effective way to develop new satellites and space crafts.

References

[1] H. Saito, and K. Hirose, “Development of Large Scale Integrated Circuit for Commercial & Space Application,” Journal of JSASS vol.58 No.683 pp.365-372, 2010.

[2] Y. Kuroda, S. Ishii, D. Takahashi, S. Kimura, H. Saito, and K. Hirose, “Development of Dual-Use Processor for Commercial & Space Application,” Journal of JSASS vol.59 No.684 pp.149-154, 2011.

[3] S. Parkes, A. Ferrer, and D. Gibson, “SpaceWire-D Standard Draft D Issue 0.15,” July 2014, unpublished.

[4] T. Yamada, “SpaceWire-R SCDHA 151-0.4 Issue 0.4,” August 2015, unpublished.

[5] “SpaceWire – Remote memory access protocol ECSS-E-ST-50-52C,” February 2010.

[6] http://www.toppers.jp/en/hrp2-kernel.html

TABLE III. Devemopment Status Of SOISOC3

Item	Status	Remark
Preliminary/Critical Design of Hardware Logic and Middleware	Complete	
Evaluation test by using FPGA	Complete	The evaluation board is exhibited at MHI booth.
ASIC#1 Design and Manufacturing	Ongoing	
Evaluation test of ASIC#1	~March 2017	
ASIC#2 Design and Manufacturing	~September 2017	Bug Fix of ASIC#1
Evaluation test of ASIC#2	~March 2018	

Software and SpaceWire evaluation of SOI-SOC3

SpaceWire components, Short Paper

Takayuki Ishida
Research and Development Directorate
Japan Aerospace Exploration Agency (JAXA)
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan
ishida.takayuki@jaxa.jp

Seisuke Fukuda, Keiichi Matsuzaki, Tadayuki Takahashi
Institute of Space and Astronautical Science (ISAS)
Japan Aerospace Exploration Agency (JAXA)
3-1-1 Yoshinodai, Sagamihara, Kanagawa 252-5210, Japan

Mitsutaka Takada, Hiroaki Takada
Graduate School of Information Science
Nagoya University
Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan

Masaharu Nomachi
Research Center for Nuclear Physics
Osaka University
1-1 Machikaneyamacho, Toyonaka, Osaka 560-0043, Japan

Takanori Narita, Masahiro Taeda, Kazunori Masukawa
Applied Electronic Equipment Design Section
Mitsubishi Heavy Industries Ltd
1200 Higashi Tanaka, Komaki, Aichi 485-8561, Japan

Keigo Saso
MHI Aerospace Systems Corp.
1200 Higashi Tanaka, Komaki, Aichi 485-8561, Japan

***Abstract*— SOI-SOC3 is a radiation hardened space-grade SOC implementing reliable SpaceWire protocol such as SpaceWire-R and SpaceWire-D. SOI-SOC3 realizes high performance in SpaceWire link speed, reliable communication technology such as data division, retransmission control, and real-time communication using time synchronization and scheduling system. These technologies are based on SOI-SOC2 technology that realized high throughput, radiation hardened, and low power consumption using commercial SOI process technology. Since such reliable SpaceWire communications are realized by various middleware, users can use reliable SpaceWire protocols just by calling specific APIs. Due to its reliability, SOI-SOC3 is applicable to not only space-grade products but also wide range of fields requiring high reliability and environment resistance such as power plants and medical devices. We have been planning to port cFE/cFS (Core Flight Executive/Core Flight System) to SOI-SOC3. Before manufacturing ASIC, we have evaluated basic function of SpaceWire on SOI-SOC3 implemented on FPGA. In this paper we describe the evaluation results of SpaceWire function of SOI-SOC3 evaluation board and its software including middleware.**

***Index Terms*— SpaceWire, SpaceWire-D, SpaceWire-R, SOI-SOC**

I. INTRODUCTION

JAXA and MHI have been developing radiation hardened semiconductor process for components in severe radiation environments. This technology achieved Single Event Latch-up free and very low probability of Single Event Upset because this technology is based on the commercial SOI (Silicon On Insulator). MHI has developed radiation hardened space-grade system-on-chip called "SOI-SOC2" using this process. This chip is mounted on a large number of satellite components. In addition to the previous SOI-SOC chip, JAXA and MHI have been developing the next generation space-grade radiation-hardened processor named "SOI-SOC3". SOI-SOC3 inherits the radiation-hard technology from SOI-SOC2. In addition to SOI-SOC2 function, SOI-SOC3 has the high-reliability SpaceWire system such as SpaceWire-D [1], and SpaceWire-R [2]. This technology makes it possible to use SOI-SOC3 in both of bus components that needs real-time communication in transfer of command and telemetry, and mission components that needs large-volume data communication in sensor data transfer. We have been developing not only processor, but also middleware and application platform using Core Flight Executive and Core Flight System (cFE/cFS) [3] created by NASA/Goddard Space Flight Center. This paper describes the outline of SOI-SOC3 and its evaluation test of SpaceWire communication function.

II. SOI-SOC3

SOI-SOC3 improved its function in both of hardware and software to implement the high-reliability SpaceWire system. In this chapter we describe the characteristics of hardware, software and middleware.

A. Hardware

To achieve short-term and steady development we employed existing technology of SOI-SOC2 such as process rule and CPU core. In addition, SOI-SOC3 has high functionality to achieve some new function below:

- Improvement in SpaceWire link rate
- Real-time communication by using time synchronization and scheduling
- High-reliability communication by data division and retransmission control

Although SOI-SOC2 can handle only RMAP as SpaceWire protocol, SOI-SOC3 can handle SpaceWire-D and SpaceWire-R as well. SpaceWire-D realizes time synchronization and scheduling function. Since control commands and sensor data are sent according to the schedule at the time of system design, SOI-SOC3 ensures real-time communication. In addition, precise scheduling and load reduction of software are realized because transmission is controlled by hardware.

SOI-SOC3 also handles SpaceWire-R protocol to realize data division and retransmission control, so reliable data transmission is allowed. Furthermore, in SpaceWire-R a specific component cannot occupy the communication because Large-sized data are divided into a fixed-sized segments.

As these high-reliability SpaceWire protocols implemented in SOI-SOC3 handle scheduling and generating headers in hardware, high throughput is achieved.

B. Middleware

To improve reusability and reduce verification scale and used application development, we constructed Application Programming Interface (API). SOI-SOC3 users are able to use SpaceWire functions easily and develop software applications using various SpaceWire protocols by calling this API.

On the other hand, SOI-SOC3 has to handle more information to use SpaceWire because SpaceWire-R and SpaceWire-D protocols are implemented. This configuration information is different between applications, so It is difficult to write all set points by users for each application. Therefore we constructed the middleware to describe each set points in static API for SpaceWire middleware that is one of the characteristics of OS (TOPPERS) used in SOI-SOC3. Middleware configurator checks the static API and constraint settings for set points, and generates necessary source code for SpaceWire communication.

C. Software

SOI-SOC3 users can use SpaceWire functions easily by calling specific API described above. We also examine porting cFE/cFS as SOI-SOC3 software platform to support necessary protocol services for user applications. Since cFE does not have interface to use SpaceWire, we are looking into adding a new service interface for SpaceWire transaction.

Figure 1 shows protocol stack of SpaceWire transaction in SOI-SOC3. SpaceWire physical layer is at the bottom, then hardware and software provide various SpaceWire functions as middleware. Users can develop software applications using these APIs.

III. Functional Evaluation

In this chapter we describe the evaluation tests of SpaceWire functions in SOI-SOC3. In this evaluation tests we evaluate basic SpaceWire function (SpaceWire-Raw).

A. Evaluation Board

Figure 2 shows the SOI-SOC3 evaluation board. All functions of SOI-SOC3 is implemented in FPGA on this evaluation board. The board has four SpaceWire ports and also has a router. In addition, one SpaceWire engine is implemented inside. This engine can send and receive simultaneously.

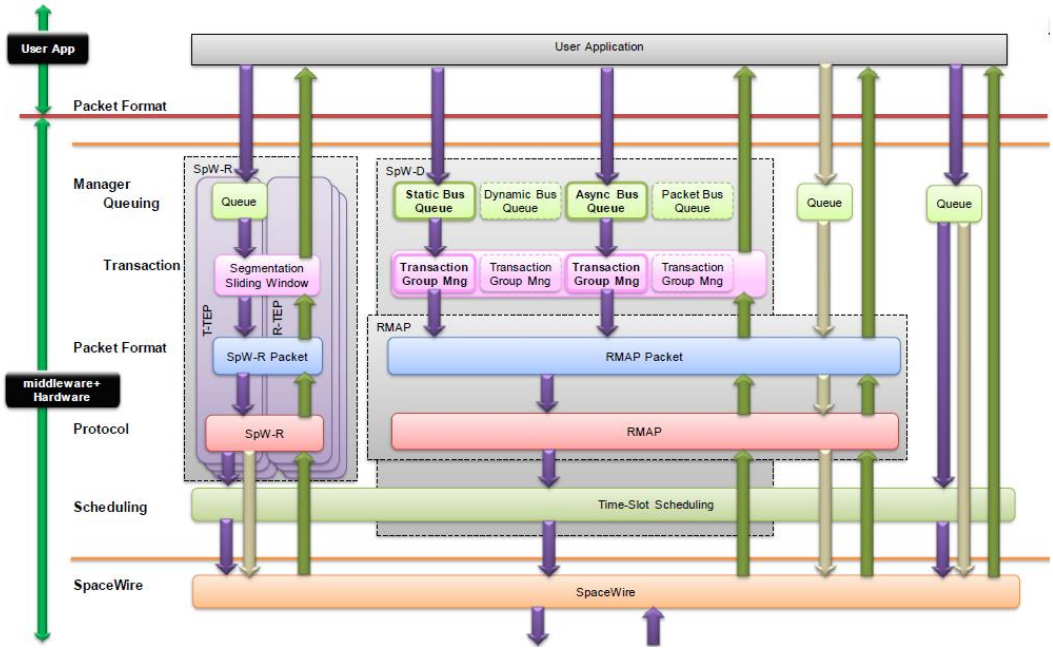


Fig. 1. Protocol stack of SOI-SOC3 [4]

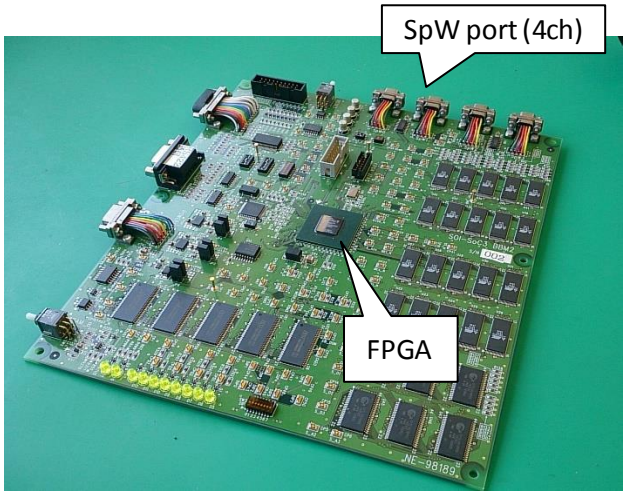


Fig. 2. SOI-SOC3 evaluation board

B. Test Environment

1) SpaceWire-Raw

Figure 3 illustrates the setup of the evaluation tests of the basic SpaceWire protocols, SpaceWire-Raw. As shown in the Fig. 3, two boards are connected via SpaceWire Link Analyzer. We measured the time to process data transaction to calculate throughput. The size of trasmit data are 1Mbyte x 8 packet and link rate is 120Mbps.

We are planning to evaluate SpaceWire-D and SpaceWire-R functions using the test environments described below.

2) SpaceWire-D

Figure 4 illustrates the setup of the evaluation tests of SpaceWire-D. In this test SOI-SOC3 #1 sends time-critical command to SOI-SOC3 #3 while SOI-SOC3 #2 sends large data which is not time-critical. The command packet can be sent as scheduled when SpaceWire-D is used.

3) SpaceWire-R

Figure 5 illustrates the setup of the evaluation tests of SpaceWire-R. In this test SOI-SOC3 #1 and #4 sends large amount of data each other via SOI-SOC #2 and #3. At the same time SOI-SOC #2 sends some packets to SOI-SOC #3. Since large data transaction shares SpaceWire path with small data transaction, large data block a small packet until its transaction ends when basic SpaceWire protocol is used. In SpaceWire-R large data are divided into some segments, therefore small packet can be sent during transaction of large data.

C. Evaluation Results

Table 1 shows the evaluation test results of SpaceWire-Raw. H/W process time means the time between the beginning of data transmission and the end of data receiving in hardware. Total process time means the time between the transmission requirement in software and the end of receiving process in software. In SpaceWire-Raw Protocol, effective throughput achieves theoretical value (96Mbps = 120MHz x 8/10 bit) . SpaceWire-D and SpaceWire-R evaluation tests will be carried out.

TABLE I. SPW-RAW EVALUATION RESULTS

Protocol	Process time [us]		Throughput [Mbps]	
	H/W	Total	H/W	Total
SpW-Raw	699077	734116	95.996	91.415

IV. CONCLUSION

We introduced next generation radiation hardened space-grade processor, which we call “SOI-SOC3”. We implemented reliable SpaceWire protocol such as SpaceWire-D and SpaceWire-R to realize high performance in SpaceWire function. We developed not only processor but also middleware to improve usability. SOI-SOC3 users can easily use various SpaceWire protocols including SpaceWire-D and SpaceWire-R by calling specific APIs. We evaluated basic functions of SpaceWire using SOI-SOC3 evaluation board and confirmed that sufficient throughput was achieved. SpaceWire-D and SpaceWire-R function evaluation tests will be carried out in near future to evaluate reliable and complicated functions.

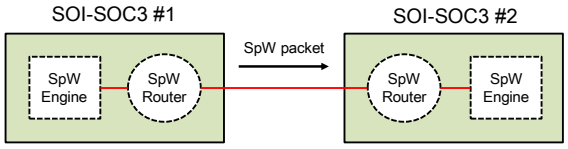


Fig. 3. Test environment for SpW-Raw

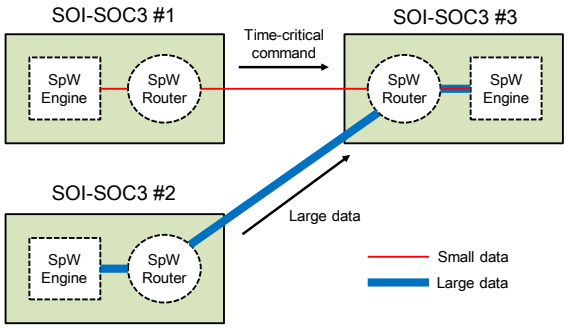


Fig. 4. Test environment for SpW-D

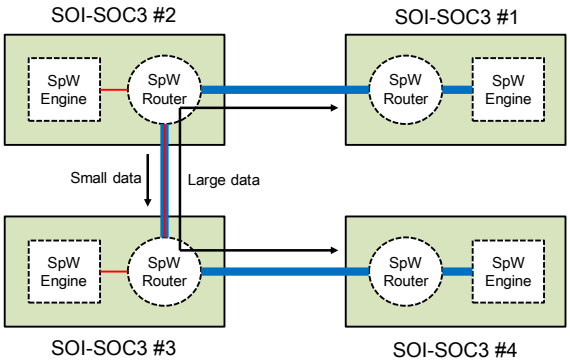


Fig. 5. Test environment for SpW-R

REFERENCES

[1] Space Technology Centre, School of Computing, University of Dundee, "SpaceWire-D, Deterministic Control and Data Delivery Over SpaceWire Networks," April 2010.

[2] T. Yamada, "SpaceWire-R SCDHA 151-0.3," Institute of Space and Astronautical Science, Japan Aerospace Exploration Agency, September 2013.

[3] D. McComas, "NASA/GSFC's Flight Software Core Flight System," Flight Software Workshop, November 2012.

[4] T. Narita, M. Kato, K. Masukawa and M. Taeda, "Development status of Next-Generation Space Grade CPU (SOI-SOC3)," 25th SpaceWire Working Group Meeting, April 2016.

Galvanic Isolation of SpaceWire Receivers

SpaceWire Components, Short Paper

G. Baterina, Y. Moghe, H. Nimmett
Silanna Group Pty Ltd
Queensland, Australia
gil.baterina@silanna.com

A. Senior
Thales Alenia Space UK Ltd
Bristol, United Kingdom
alan.senior@thalesaleniaspace.com

***Abstract*—This paper summarizes the need for galvanic isolation in SpaceWire networks and reviews the limitations of current isolation solutions; the paper also proposes a new Isolated SpaceWire Receiver device based on radiation-hardened Silicon-on-Sapphire (SoS) technology and demonstrates the benefits of primarily isolating only the receiver lines of an SpW port. As a stepping-stone to the proposed device, a discrete galvanic isolation receiver module will be demonstrated using Silanna's core isolator chip along with commercial off-the-shelf (COTS) LVDS transmitters, LVDS receivers, and an isolated DC-DC converter to power all circuitry across the isolation barrier. To ease testing & evaluation, the SpW isolator module can be simply implemented by cascading inline to an existing SpW port, or by replacing the SpW pigtail connector, or even by replacing the LVDS transmitter & receiver stages. The proposed integrated solution is expected to have on-chip isolated power and operate up to 400 Mbps, handling a common-mode of 100 V-RMS, and a galvanic isolation of 1 kV-RMS.**

***Index Terms*— SpaceWire, SpW, isolation, fault propagation, LVDS, common mode voltage, galvanic, component, spacecraft electronics.**

I. Introduction

SpaceWire (SpW) as defined in the standard [1] uses the Low Voltage Differential Signaling (LVDS) electrical interface which has the advantage of reducing the power required for a high speed data link; however, the existing LVDS buffers and ASIC devices have 2 principal drawbacks for implementing high reliability systems:

- limited common mode voltage tolerance
- fault propagation paths

The common mode tolerance is +/-1V; if this voltage is exceeded then the link data may be corrupted. In the worst case the transmitter/receiver devices may either be stressed or permanently damaged. Stressing of the LVDS buffer may not be evident but often results in a reduced reliability leading to premature failure later. Within a spacecraft, it is practical to control the common mode voltages within the specified limits and thus once launched problems would not be anticipated. Control of the common mode voltages during ground testing of spacecraft with remote Electrical Ground Support Equipment (EGSE) that use long cables becomes more problematic; drivers and receivers have failed in test configurations either due to incorrect test setups, poor grounding setups, or the effects of EMC testing. Clearly it is important to implement an effective grounding scheme and ensure that methods for monitoring the common mode voltages are in place rather than wait for failures to occur or assume acceptable conditions are met.

Fault propagation paths exist between LVDS link ends due to the direct silicon to silicon connection between the devices at the two ends of a link [2]. As shown in Figure 1, a power supply failure in one piece of equipment could propagate to another equipment by injecting out of specification voltages at the LVDS buffer terminals [3]. Due to the constraints of high speed signaling, it is not practical to use series protection resistors in the signal lines to reduce potential fault currents to an acceptable level; thus, it is necessary to add protection to the internal supply rails of each equipment.

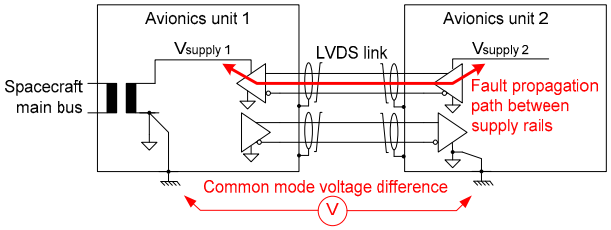


Fig. 1. *Common mode voltages and fault propagation*

The mitigation methods for both the common mode and fault propagation issues are time consuming to analyze for failure mode effects and they typically result in an increased complexity of the flight equipment.

It is thus highly desirable to incorporate galvanic isolation in the link paths; this will permit the legacy Mil-Std-1553B command and control links to be replaced with the more capable SpW bus and to eliminate failures in test environments with EGSE. [4]

II. Limitations of Current Isolation Solutions

The Data and Strobe lines of SpaceWire are non-DC-balanced signal streams with data rates up to 400 Mb/s. [1] Since the signal streams are not DC-balanced, typical capacitive or inductive (transformer) AC coupling methods for isolation are not viable; in comparison, by design, high speed

digital isolators are capable of handling non-DC-balanced signal streams. However, almost all high speed digital isolators available today have maximum data rates of less than 200 Mb/s; although there are some digital isolators capable of data rates greater than 200 Mb/s, they are not built on a space-proven, radiation-hardened process such as Silicon-on-Sapphire (SOS). Silanna has already demonstrated the digital isolation of signal streams greater than 500 Mb/s using a 0.5µm SOS process. [5]

III. Limitations of Previously Proposed SpaceWire Link Isolator

The initial proposal for SpW isolation was a SpaceWire link isolator that fully isolated both the transmitter pair (Data and Strobe) and the receiver pair of LVDS interfaces. [4] While this approach had successfully provided isolation to SpW links operating up to 400 Mb/s, there were some issues when both ends of a SpW connection had link isolators; with the component sides of both links establishing their own separate ground references, the fully isolated cable resulted in a "floating" ground reference for the signal levels within the cable. This represented a potential problem of exceeding the isolation voltage range of one or both SpW link isolators.

Although the floating cable ground problem could be mitigated by ensuring only one end of a SpW connection is fully isolated, this complicates the SpW network design of a system.

IV. Proposed Isolated SpaceWire Receiver

To prevent the potential problems of having both ends of a SpW connection fully isolated, the proposed Isolated SpW Receiver would have two high-speed data channels to handle Data and Strobe of the receive signaling only; with LVDS levels on the cable-side and LVDS & LVTTL levels supported on the module-side of the isolation barrier, both discrete and integrated SpW ports could have isolated receivers with a nearly drop-in isolation solution. To further simplify the adoption of the Isolated SpW Receiver, the device would also include the integration of a DC-DC isolator to optionally provide power to the cable-side of the receiver from the module-side without the need for additional active components (see Fig. 2).

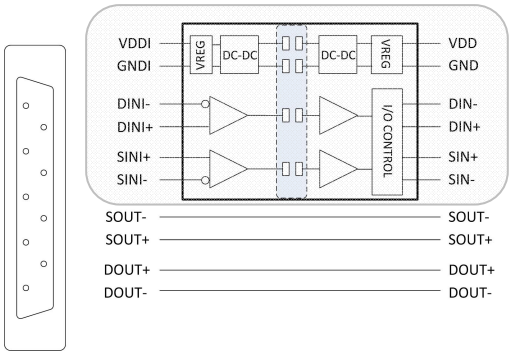


Fig. 2. *Silanna Isolated SpW Receiver*

A summary of the target features are:

- 2 high speed (400 Mbps) channels
- Cable-side: LVDS Inputs
- Module-side: LVTTL or LVDS Outputs
  - LVTTL: Receiver Mode
  - LVDS: Repeater Mode
- LVDS failsafe per SpW standard
- Cold sparing for redundant backup
- Isolation voltage: 1 kVrms
- Working voltage: 100 V (common mode voltage)
- Integrated DC-to-DC isolator to power cable-side from module-side
- Cable-side receiver lines align well w/ SpW cable receiver connections
- Ground-based device in 12-pin plastic package
- Space grade device in 12-pin ceramic package
- Silicon-on-Sapphire (SOS) technology
- Target Radiation Tolerance > 100 krad(Si) TID (for Space grade)

V. Benefits of Receiver-only SpW Isolation

In addition to being smaller, lighter, and consuming less power compared to a fully isolated SpW port, the Isolated SpW Receiver allows greater flexibility in configuring a SpW network and localizing isolation to the ports, nodes, and modules that are required to handle higher common-mode voltages. The floating ground within the SpW cable is avoided since the signaling is referenced to the transmitter grounds on either end of the cable.

VI. Demonstration Module

To demonstrate the high speed digital isolation capabilities in a SpW application, a demonstration (demo) module for the Isolated SpW Receiver was designed around the Silanna SIL1020L 2-channel high-speed digital isolator device (Fig. 3). A dual channel LVDS receiver (LV028 type) is used on the cable-side of the isolator while a dual channel LVDS transmitter (LV027 type) is used to interface with the module-side. An isolated DC-DC converter is also included as an option to power the isolated side of the module. Prototypes of the module demonstrated the wide common mode range of the isolated interface and the capability to handle the non-DC-balanced pseudo-random data streams up to 400 Mbps.

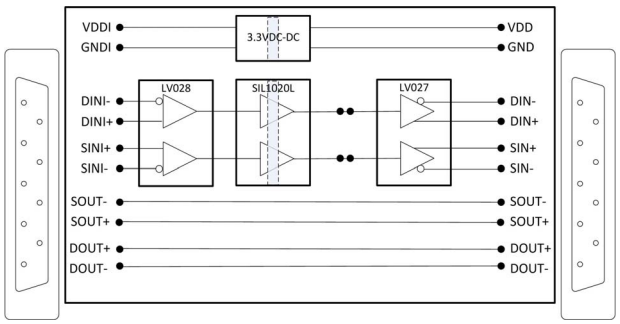


Fig. 3. *Isolated SpW Receiver Demo Module*

With the appropriate SpW connectors on both ends of the demo module, a SpW port's receiver can be easily isolated and tested by simply installing the demo module between the SpW cable & the SpW connector and supplying power (3.3V) from the target system. Optionally, SpW connector pigtail wires can be used on the powered-side of the module to replace a standard pigtail connector for testing. There are also connection points between the SIL1020L high-speed digital isolator and the LVDS transmitters (LV027) to allow replacement of the existing LVDS drivers & receivers where applicable.

Testing of the Isolated SpW Receiver demo module will continue with tests in SpW environments that include both AC and DC common mode voltages.

VII. Conclusions

The isolation of the receivers in a SpW port addresses the limited common mode voltage range of standard SpW connections. With an Isolated SpW Receiver both ends of a SpW connection, there will not be a direct connection of the devices at the two ends of the link; this severs the fault propagation path between the two devices that could occur in the event of a power supply failure.

Additional testing at Silanna and within the SpaceWire community is needed to confirm the effectiveness of isolating only the receivers of a SpW port.

.

References

[1] European Space Agency - ECSS Secretariat, "ECSS-E-ST-50-12C, Space engineering, SpaceWire – Links, nodes, routers and networks," 31st of July 2008, 129 pages

[2] M. Suess, J. Ilstad, W. Gasti, "Galvanic Isolated SpaceWire Links, Requirements, Design Options and Limitations," 2009 ESA Workshop on Reliable Power & Signal Interfaces

[3] R. Malmberg, "Failure Propagations via Power and Signal Interfaces," 2009 ESA Workshop on Reliable Power & Signal Interfaces

[4] G. Baterina, Y. Moghe, P. Francois, and A. Senior, "Galvanic Isolation of SpaceWire Links," International SpaceWire Conference 2013

[5] Y. Moghe, A. Terry and D. Luzon, "Monolithic 2.5kV RMS, 1.8V - 3.3V Dual-Channel 640Mbps Digital Isolator in 0.5µm SOS," SOI Conference (SOI), 2012 IEEE International, On page(s): 1 - 2

90 nm 12.5 Gbit/s physical interface per SoC with SpaceFibre/GigaSpaceWire links for the space radars

Components, Short Paper

Dmitri Skok, Tatiana Solokhina, Jaroslav Petrichkovich, Juri Gerasimov
ELVEES RnD Center,
Zelenograd, Russia,
tanya@elvees.com

***Abstract* — The article presents 12.5 Gbit/s Physical Media Attachment (PMA) units, TX and RX, fabricated in 90 nm bulk CMOS process. The PMA are designed for use in SpaceFibre/GigaSpaceWire (SpaceWire-RUS) systems for the space radars. The units comprise SERDES and clock and data recovery (CDR). Supported set of data rates includes those of 1.25, 2.5, 6.25 and 12.5 Gbit/s, but intermediate rates are also available.**

***Index Terms* — SoC, space components, SERDES, PMA, SpaceFibre, GigaSpaceWire.**

I. INTRODUCTION

The data transmission systems become more and more demanding in terms of throughput. Driving applications of the high-speed links include uncompressed video transmission and wideband radio applications, including radars. SpaceFibre and GigaSpaceWire networks are attractive solutions due to their elaborate networking capabilities, though their throughput yet can be a bottleneck for certain applications. This work describes the integrated solution to achieve channel rates of up to 12.5 Gbit/s per link using 90 nm bulk CMOS process.

II. MOTIVATION

As the bandwidth processed by the interface SoC increases, datarate grows proportionally. This requires either more pins, or more bandwidth per pin. Since the pin count increase is not always possible or desirable for space and parasitics reasons, one is seeking for higher throughput per pin. SpaceFibre is a good choice for several reasons. Unlike protocols like Jesd204b, it can be used both for bulk data transfer and configuration, an also has extensive networking capabilities.

III. ARHITECTURE

The original application of the work is the SoC for phased array radars. GigaSpaceWire was selected as the data interface due to its ability to support both system configuration and monitoring and high-speed bulk data transmission. Also, the protocol is relatively lightweight in terms of silicon area.

The SoC can process radio data in the bandwidth as wide as 600 MHz. Signal received is digitized by the 12-bit quadrature ADC. Sample rate after filtering and decimation is 700 MSa/s, giving the total I/Q payload data rate of 16.8 Gbit/s. After 8b10b expansion this becomes 21 Gbit/s, not counting service traffic. If the data is formatted as 16 bit words instead of 12 bits, the total rate required increases to 28 Gbit/s.

To alleviate this bandwidth requirements, several techniques were used. Among them bit packing, block-floating point representation, signal bandwidth limiting.

Nonetheless, at least one 12.5 Gbit/s link is required to transmit useful amount of data, while 2x is needed for full bandwidth.

There are four GigaSpaceWire physical interfaces per SoC, that can be configured as 4*1x, 2*2x or 1*4x.

Despite this high data rate, the distance requirements are not very high in this application. Since the data are processed collectively by units, only the links to the neighbors are required. Typically, there are few centimeters of PCB.

To address the speed requirements, the corresponding TX and RX PMA units were designed. The link speed is up to 12.5 Gbit/s. From the internal side, data is fed over 4*10 bit bus at the clock rate of up to 312.5 MHz.

The TX unit performs data serialization. Internally, data are fed by the 40-bit parallel interface, 4 symbols per clock. Reference clock may by any from 200 MHz to 600 MHz in 50 MHz increments. For 12.5 Gbits/s the minimum clock frequency is 350 MHz.

The output provides typical 400 mV p-p differential into 100 ohm load. It also provides high frequency pre-emphasis [1].

The RX unit performs clock-and-data recovery and data deserialization, based mostly on [2]. It also has 40-bit data interface and similar clock specification. The data alignment and error detection are performed in the digital domain.

The unit does not perform adaptive equalization.

IV. PROOF ON SILICON

To test end evaluate the PMA modules, test chips were fabricated. The process is TSMC bulk CMOS 90 nm, 7 metal layers.

To feed the TX unit with the 12.5 Gbit/s data over external CMOS IO pins, one would need 100 of them at 125 MHz clock, and the same number for RX. This would require 200+ pins package. Such package would have considerable size, hence, parasitics, and cost.

To keep the pin count low, we placed essential data pattern generation and checking logic on-chip, accessible over serial interface. This allowed us to fit the chip, containing also other test structures, into a compact 64-pin QFN package.

The logic provides the following features and functionality:

- configuring TX and RX units for various bit rates and reference clock frequencies;
- sending either pre-defined or user-programmable 40-bit patterns;
- sending pseudo-random legal 8b10b sequence;
- capturing 40-bit words received;
- counting of total words received;
- counting of bit errors when transmitting static 40-bit pattern into Bit Error Counter (BEC);
- counting bit errors when transmitting pseudo-random sequence into BEC;
- automatic capturing the first word, containing bit error.

The test fixture is interfaced with PC over USB.

The PC control and monitor software allows easily perform operations like atomically setting and clearing individual bits in the pattern transmitted, automatic and manual alignment, pattern shifting, real-time BER display, RX data rate, etc.

In the first mode of operation, one can send arbitrary static pattern and observe the one received, (fig. 2).

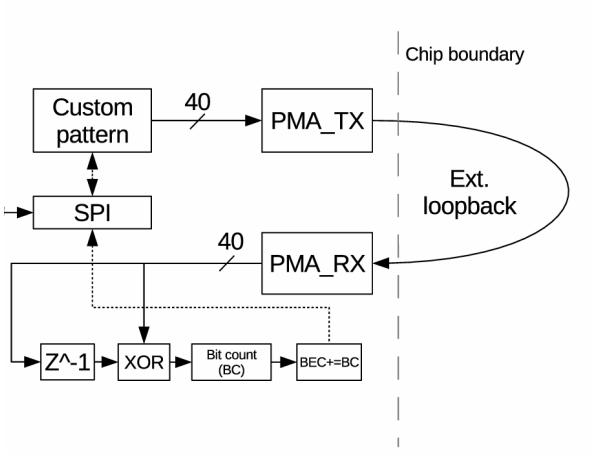


Fig. 2. Static pattern testing.

In this mode, bit errors are counted as number of bit difference between successive two words received. For instance, flipping any bit in TX pattern would increment BEC by one. This approach would underestimate BER when some bit is received erroneously several times in sequence. Nevertheless, on the first occurrence, the error would be counted, and the word containing it would be captured for analysis.

If the pattern transmitted contains COMMA, the RX would align automatically once. It can then be reset for realignment, if necessary.

In the second mode, the TX initially sends starting sequence, containing COMMA in every 4-th symbol. RX aligns to that sequence and goes to the “READY” state. When the TX then eventually switches to sending pseudo-random sequence, RX notices the pattern change and starts to compare the sequence received with that of the local pseudo-random generator. Thus, every bit error can be counted in BEC and the true BER measured (see fig. 3).

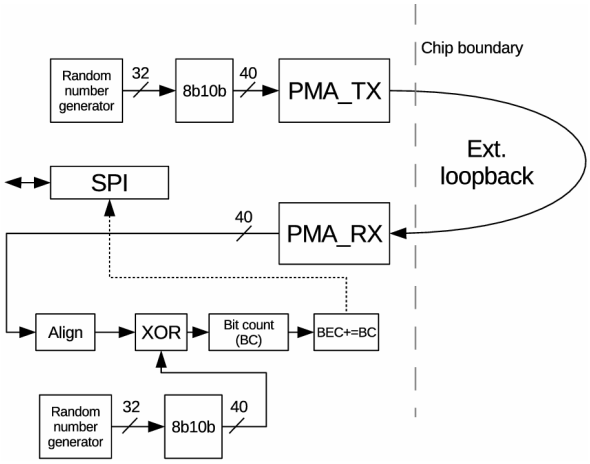


Fig. 3. Random sequence testing.

Two types of PC-boards were fabricated, one for local loopback (see Figure 1), and another for wired board-to-board

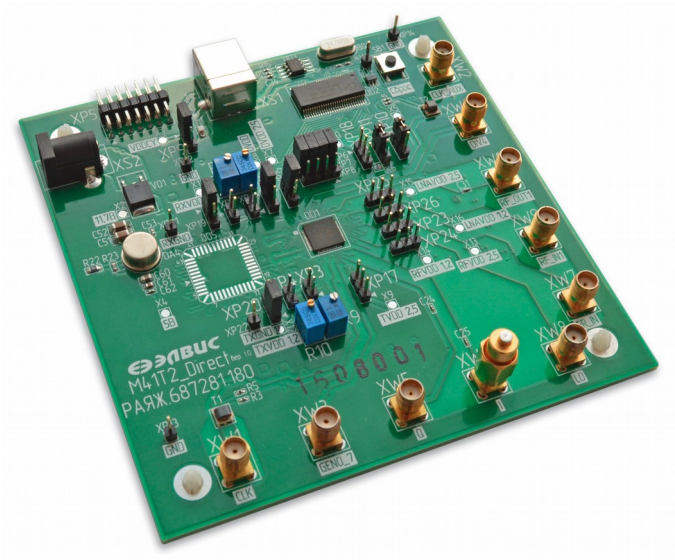


Fig. 1. PC board with local loopback.

or wired loopback configuration. In the latter case, a SATA cable with appropriate connectors was used.

V. Results.

Test chips were tested for speeds 1.25 Gbit/s, 2.5 Gbit/s, 6.25 Gbit/s and 12.5 Gbit/s.

Samples show reliable communication at rates 1.25 and 2.5 Gbit/s over local loopback and board-to-board over 30 cm

cable showing BER<10^{-13} (no bit errors detected during 3 hour testing).

For the two higher rates results are varying. Local loopback shows BER of about 10^{-11}at 6.25 Gbps and 10^{-9} at 12.5 Gbps.

Over 10 cm cable BER is 10^{-10} at 6.25 Gbps and 10^{-5} at 12.5 Gbps. With 30 cm cable BER is still below 10^{-7} at 6.25 Gbps, but at 12.5 Gbps the communication is unreliable.

VI. Conclusion

The implemented and tested PMA subsystem supports link rates of up to 12.5 Gbit/s per link. Test structures were fabricated in 90 mn bulk CMOS process and tested.

The units are suitable for chip-to-chip communication within PCB at speeds up to 12.5 Gbps.

For use over longer cable lines, RX equalization would be required.

References

[1] Jin Liu, Xiaofeng Lin, “Equalization in high-speed communication systems”, IEEE Circuits and Systems Magazine Year: 2004, Volume: 4, Issue: 2

[2] Jri Lee, B. Razavi, “A 40-Gb/s Clock and Data Recovery Circuit in 0.18-um CMOS Technology,” IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 38, NO. 12, DECEMBER 2003.

Innovative miniaturization for low resource interplanetary exploration

Components, Short Paper

Seisuke Fukuda, Takayuki Ishida
Institute of Space and Astronautical Science (ISAS)
Japan Aerospace Exploration Agency (JAXA)
Sagamihara, Chuo-ku, Kanagawa, Japan
{fukuda@isas., ishida.takayuki@}jaxa.jp

Hiroki Hihara, Yoshinori Matsuo,
Mitsunobu Kuribayashi, Hiroshi Matsushima
Space Engineering Division
NEC Space Technologies, Ltd.
Tokyo, Japan
{h-hihara@bc, y-matsuo@pi, m-kuribayashi@cp,
h-matsushima@wx}.jp.nec.com

Takahiko Tanaka, Osamu Watanabe
Space Systems Division
NEC Corporation
Tokyo, Japan
{t-tanaka@dy, o-watanabe@ak}.jp.nec.com

Koichi Shinozaki, Toshiyuki Yamada
Aerospace Research and Development Directorate
Japan Aerospace Exploration Agency (JAXA)
Tsukuba, Ibaraki, Japan
{shinozaki.koichi, yamada.toshiyuki}@jaxa.jp

***Abstract*— The Japan Aerospace Exploration Agency (JAXA) launched new research aiming at realizing low resource and frequent space science mission. We report the development activity for realizing miniaturization of onboard equipment using SpaceWire. The technology development target is the miniaturization of onboard units, which are small enough for deploying interplanetary mission with small rockets as Epsilon Launch Vehicles. Since SpaceWire interface consists of full digital circuitries and its communication protocol enables hardware implementation, Large Scale Integration (LSI) technology, surface mounting technology (SMT), and ceramic ball grid array (CBGA) packages are exploited for the miniaturization of onboard equipment with communication interfaces. The preliminary result shows that one half and/or one third in scale and mass can be realized.**

***Index Terms*— SpaceWire, Surface Mounting Technology (SMT), Interplanetary Mission, Miniaturization.**

I. INTRODUCTION

JAXA launched new research aiming at realizing low resource and frequent space science mission. The technology development target is the miniaturization of onboard equipment, which is small enough for deploying interplanetary mission with small rockets as Epsilon Launch Vehicles. Since SpaceWire interface consists of full digital circuitries and its communication protocol enables hardware implementation, it plays an important role for the miniaturization of onboard equipment with communication interfaces. We report the preliminary result of our development activity, in which one half and/or one third in scale and mass of onboard equipment can be realized with SpaceWire interfaces.

The key technologies are miniaturization using Large Scale Integration (LSI), surface mounting technology (SMT), and ceramic ball grid array (CBGA) packages. LSI technology is the straight forward method for downsizing, and a fully SpaceWire based satellite has been successfully demonstrated on orbit by HISAKI [1, 2], the extreme ultraviolet spectroscope for Exospheric Dynamics, launched in 2013 and is working in stable condition. All onboard bus equipment of HISAKI employs SpaceWire interfaces, and scaling law with LSI technology is observed. CMOS (complementary metal-oxide semiconductor) LSI scaling law is also inherited for the downsizing of onboard equipment. SpaceWire interface plays an important role on HISAKI, which realizes scaling law of miniaturization for straight forward adoption of LSI technology.

We also report the evaluation result of environment tests of miniaturization technology such as surface mounting technology using versatile CBGA packages. These items are applicable for various LSI dies.

The miniaturization of SpaceWire based onboard equipment enlightens downsizing of other subsystems such as thermal control, structure, power unit, etc. These improvements are expected to accelerate the downsizing of whole satellite system.

II. MINIATURIZATION DEMOSTRATED BY HISAKI

Spectroscopic Planet Observatory for Recognition of Interaction of Atmosphere "HISAKI" (SPRINT-A) is the world's first space telescope for remote observation of the planets such as Venus, Mars, and Jupiter from the orbit around the earth. Every bus equipment of HISAKI employs SpaceWire for communication interfaces. Its attitude and orbit control

subsystem (AOCS) has standby redundancy, whereas the mass is only 348kg and the bus size is small as shown in Fig. 1 with the payload of Extreme-ultraviolet (EUV) imaging spectrometer.

©JAXA

Fig. 1. HISAKI (SPRINT-A) with a system test crew member

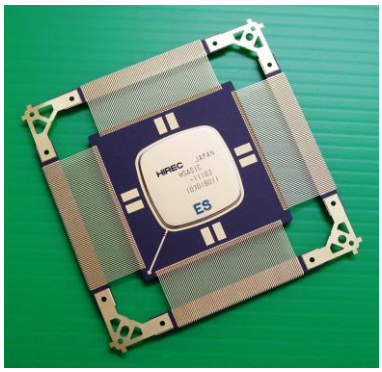
Miniaturization of HISAKI has been achieved by LSIs with SpaceWire interfaces. All communication LSIs incorporate SpaceWire interfaces as shown in Fig. 2, and they contributed the reduction of size and mass of bus equipment of HISAKI.

NSR14 is a SpaceWire router with 20 ports, on which 14 physical ports and 6 virtual ports are implemented. It is shown in Fig. 1 (a). It is fabricated on JAXA authorized 0.15 μm Silicon-on-Insulator (SOI) cell-based application-specific integrated circuit (ASIC). Two 28-port SpaceWire routers are used in HISAKI, and one SpaceWire router accommodates 2 NSR14 LSIs.

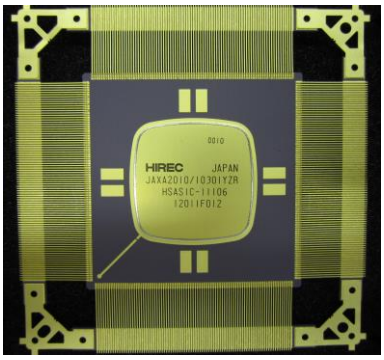
Multi-mode Intelligent Terminal (MIT) shown in Fig. 2 (b) is an input/output processor (I/O processor) with an 20-port SpaceWire router and 2 communication micro-controller units (MCUs), which is also fabricated on JAXA authorized 0.15 μm SOI cell-based ASIC. One MIT is used in Space Cube2 onboard computer in order to implement SpaceWire-D [3] for guaranteeing determinism [4, 5].

Network Interface Controller (NIC) is a terminal function controller LSI with the target function of Remote Memory Access Protocol (RMAP) [6]. The first version of NIC is NIC07, which is shown in Fig. 2 (c), and it is fabricated JAXA authorized 0.35 μm CMOS cell-based LSI. The circuitry implemented in NIC07 is prepared as an Intellectual Property (IP) in order to accommodate its functions in Field Programmable Gate Arrays (FPGAs) for the interface modules of sensors, actuators, power control subsystem equipment, heater control electronics units, and telecommunication equipment.

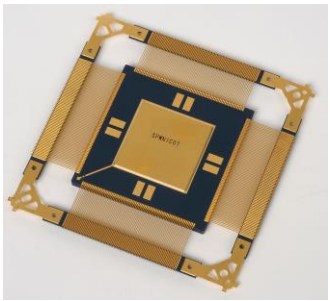
All of the design was described in high level language as ANSI-C language using ELEGANT framework [7] in the preliminary design phase, in consequence the verification time scale was shorter than the design process using Register-Transfer Level (RTL) like Verilog or VHDL (VHSIC (Very High Speed Integrated Circuit) Hardware Description Language). Therefore, these SpaceWire communication LSIs were intended to contribute the miniaturization of onboard equipment rather than aiming at developing a standard SpaceWire communication LSI.



(a)



(b)



(c)

© JAXA

Fig. 2. SpaceWire communication LSIs (a) NSR14: 20-port SpaceWire router, (b) MIT: Multi mode Intelligent Terminal with an 8-port SpaceWire router and two communication micro-controller units, (c) NIC07: Network Interface Controller with SpaceWire/RMAP terminal functions.

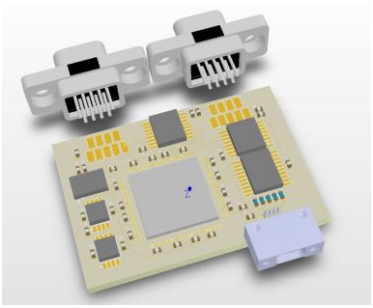
III. Surface Mounting Technology for Miniaturiation

Since the miniaturization development with LSIs with SpaceWire interfaces were successfully demonstrated by HISAKI, JAXA has started the next step of miniaturization. Major technology development activities are SMT and providing versatile CBGA packages.

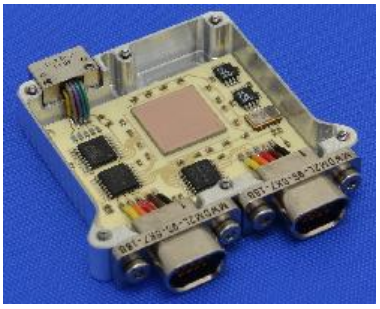
A. Surface mounting technology (SMT)

SMT is the most prospective technology for the miniaturization of protocol bridges between SpaceWire and legacy interfaces. When we incorporate legacy interface devices used for sensors and actuators, we should integrate mixed signal LSIs, analog Integrated Circuit (IC) as well as logic ICs and LSIs. Hybrid IC (HIC) and multi-chip module (MCM) are candidates for integrating those mixed signal devices in one package, whereas the packages of HIC/MCM are obstacles for achieving substantial miniaturization. We aim at miniaturized units without HIC/MCM packages instead.

The miniaturization development activity is in concept design phase, in that we have collaboration between overseas partners in order to realize smart sensors and smart actuators for integrated onboard networks with SpaceWire. We selected an interface module with a SpaceWire interface for converting the legacy interfaces of sensors like GAS (Geomagnetic aspect sensor), CSAS (Coarse sun aspect sensor), and SPSH (Sun Presence Sensor Head) as the motif of the first step, because it has typical mixed signal interfaces.



(a)



(b)

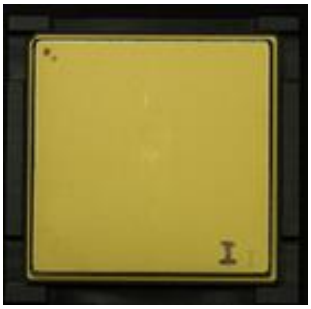
The result of Europe and Japan collaboration

Fig. 3. A mockup of miniaturized SpaceWire bridge

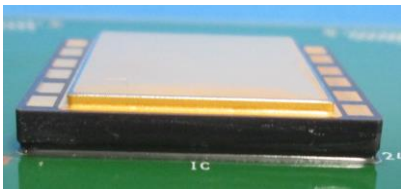
Figure 3 shows the mockup of concept design derived through the collaboration between European and Japanese members. Figure 3 (a) shows the design concept of the SpaceWire bridge for a legacy interface, and Fig. 3 (b) shows a mock up using real size components with the same functions as flight devices. Two A5 size modules are expected to be shrunk into the size of a connector back-shell. What we found is that the smaller the module is, the less numbers of passive electronic components and mechanical parts are required.

B. Versatile Ceramic Ball Grid Array (CBGA) package

BGA packages are indispensable components for miniaturization. The issue of the package is that a BGA package and/or a CGA (Column Grid Array) are provided for each device, and each device vendor has to bear non-recurring cost for each package.



(a)



(b)

© *JAXA*

Fig. 4. JAXA authorized CBGA package, (a) top view, (c) side view

TABLE I. CBGA PACKAGE LINEUP

Item	**Spaceborn CBGA package lineup**		
size, pin numbers, terminal pitch	26 x 26 mm, 572 pin, 1.0 mm pitch	21 x 21 mm, 357 pin, 1.0 mm pitch	15 x 15 mm, 165 pin, 1.0 mm pitch
Cavity	4 steps	4 steps	2steps
Soldering	High melting point dimple solders SnPb soldering for terminals		

Figure 4 shows the top view (a) and side view (b) of the JAXA authorized CBGA 572 pin package. Three types of the CBGA packages are to be provided as shown in Table 1. These packages are provided with JAXA authorized assembly criteria document, and various kinds of chips of devices can be mounted inside the CBGA package. The environmental evaluation based on JAXA authorized test condition is successfully carried out.

IV. Electrical design evaluation

Electrical design evaluation follows the consideration of assembly technology reported in the preceding section. The issue of the evaluation is to find out how assembly technology contribute to the minimization of electrical design. We selected a SpaceWire bridge for RS-422 interface as the second motif, which is called Payload Interface Unit (PIU). The typical flight model of the bridge unit is used on orbit in order to attach a conventional Global Positioning System Receiver (GPSR) to an onboard satellite bus within SpaceWire network, and is shown in Fig. 5 and Table II.

PIU: RS-422 to SpaceWire bridge

Fig. 5. Payload interface unit (PIU) for GPSR

TABLE II. Payload interface unit (PIU) for GPSR

Supported sensor	GPS receiver
Size (mm)	142 (W) x 150 (D) x 81.4 (H)
Weight (kg)	1.46
Power (W)	6.51 (typical)

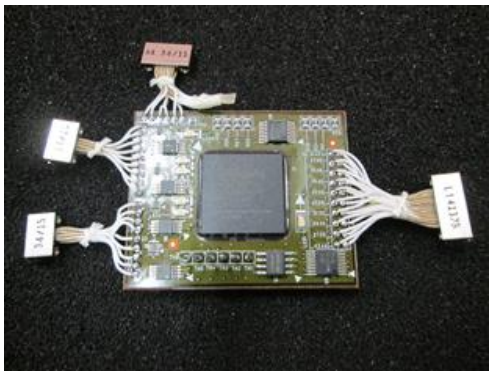
The original PIU has its own power supply unit (PSU) for providing secondary power, and two PWBs for digital function circuitry.

We exclude the PSU for evaluating minimization design in order to set up a scope on minimization of digital circuits. The electrical functions of the miniaturized module are verified, and the module has the same function as the original PIU. We found that the miniaturization of a unit is difficult from the bottom-up design consideration of minimization through this evaluation. The reduction of passive electronics devices and interface devices are taken into consideration from the top-down point of view encouraged by the preceding assembly technology evaluation. The verified module with a surface mounted FPGA using surface mount technology is shown in Fig. 6. Two radiation hardened FPGAs are used in the flight model of the PIU. They are ACTEL FPGAs RTAX2000S-CQ352. The whole functions on two FPGAs are implemented on one commercial FPGA with similar system gate size. We selected one Spartan-6 XC6SLX25-3FT256 for our evaluation, because its capacity is close to the FPGAs for the flight unit as shown in Table III. Some flight level FPGAs found to be suitable for the flight model of this one chip implementation, because the elimination of interface circuit between two FPGAs is noticeable for the reduction of system gate size.

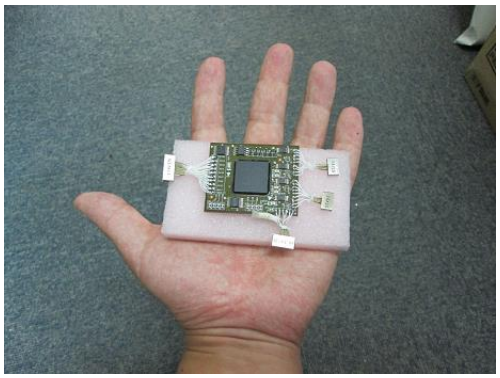
TABLE III. Evaluation model FPGA

Item	Flight Model	Electrical evaluation model
FPGA	RTAX2000S-CQ352	Spartan-6 XC6SLX25-3FT256
Flip-Flops	10,752 x 2	54,576
Memories	288 kbits x 2	2,088 kbits
I/Os	198 pins	186 pins

The target size of the miniaturized module is expected as 39 mm x 48 mm x 10 mm at first with the premise that whole circuit over the two PWBs can be transposed into one FPGA. Once the target size is established, the reduction of the numbers of passive electronics devices and interface circuitries is taken into account.



(a)



(b)

© JAXA

Fig. 6. Payload interface unit (PIU) for GPSR

V. Conculusion

JAXA's next generation miniaturization technology development activity is introduced. The premise of reduction in size is based on the evaluation of assembly technology and

versatile JAXA authorized CBGA packages. The perspective of miniaturization boosts the simplification of electronics design.

REFERENCES

[1] S. Fukuda, "Small Bus Technology for Scientific Satellites - Accomplishments of HISAKI/SPRINT-A and Future Perspective -," IEICE Technical Report, SANE2014-12, May 2014.

[2] K. Nakaya, S. Fukuda, S. Sakai, A. Yamazaki, K. Uemizu, T. Toriumi, J. Takahashi, M. Maehara, T. Okahashi and S. Sawai, "Development of Flexible Standard Bus for ISAS/JAXA Small Scientific Satellite Series," Trans. JSASS Space Tech. Japan, vol.10, no.ists28, pp.Tf_5-Tf_9, 2012.

[3] Space Technology Centre, School of Computing, University of Dundee, "SpaceWire-D, Deterministic Control and Data Delivery Over SpaceWire Networks", April 2010.

[4] T. Takahashi, T. Takashima, S. Kuboyama, M. Nomachi, Y. Kasaba, T. Tohma, H. Hihara, S. Moriyama, T. Tamura, "Space Cube 2 - An Onboard Computer Based on Space Cube Architecture, " International SpaceWire Conference 2007, p.65-68, September 2007.

[5] H. Hihara, T. Ogawa and K. Kitade, "NEXTAR: Small Satellite Bus Based on SpaceWire Deterministic Implementation," International SpaceWire Conference 2011, p.344-347, November 2011.

[6] European Space Agency, ECSS-E-ST-50-52C, "Space engineering, SpaceWire - Remote memory access protocol", February 2010.

[7] H. Hihara, Y. Nishihara, M. Nomachi, T. Takahashi, and T. Takashima, "Designing Space Cube 2 with ELEGANT Framework, " International SpaceWire Conference 2008, p.219-222, November 2008.

Wednesday 26 October

Missions & Applications (Long)

Common SpaceWire Software for ESA JUICE Instrument Payloads

SpaceWire Missions and Applications, Long Paper

Martin Åberg, Daniel Hellström, Arne Samuelsson, Felix Siegle, and Sandi Habinc
Cobham Gaisler AB
Gothenburg, Sweden
[maberg | daniel | arne | felix | sandi] @gaisler.com

Felice Torelli
European Space Agency
Noordwijk, The Netherlands
felice.torelli@esa.int

***Abstract*—The common DPU platform for ESA JUICE mission instruments is a hardware and software platform developed by Cobham Gaisler for the scientific instrument payloads of the European Space Agency Jupiter Icy Moons spacecraft. The hardware is based around the GR712RC dual-core LEON3-FT processor with GRSPW2 SpaceWire interfaces. To accompany the JUICE instrument hardware, a flight quality SpaceWire software package has been developed, compliant with ESA ECSS standards for Space Software engineering. The software includes SpaceWire device drivers and protocol support for the SpaceWire CCSDS Packet Transfer Protocol, the Packet Utilization Standard and the SpaceWire Time Distribution protocol.**

***Index Terms*—JUICE, Scientific Instrumentation, SpaceWire**

I. Introduction

Defining a common hardware and software platform for computer systems in a spacecraft is difficult as space missions and their instruments are dedicated to perform a specific task optimized for weight, power, performance and many other parameters. Despite differences, instruments may have many commonalities such as operating conditions, radiation environment, criticality and on-board network communication protocols and more.

As part of the ESA funded activity "DPU for JUICE Instruments" contract 4000113396/15/NL/BW, Cobham Gaisler designed a DPU (Digital Processing Unit) hardware and software platform to meet the common requirements of ten payloads on-board the JUICE satellite [1]. ESA had compiled functional and performance requirements and demanded components with flight heritage, radiation tolerant up to 100krad(Si) total dose, configurable in the performance range up to 100 MIPS and working memory up to 256MiB. Cobham Gaisler proposed adaptations and realized the requirements into a platform based on the GR712RC LEON3-FT dual-core processor [2] described hereafter.

During the activity boot and driver software supporting the DPU platform was to be developed, unit-tested, validated and documented according to project specific tailoring of the ECSS-E-ST-40C and ECSS-Q-ST-80C software engineering and quality standards [3], [4].

The software package consists of a boot loader, Standby Mode remote maintenance software and a hardware driver library. The boot follows the ESA flight software boot loader payload requirements [5] and the driver software was designed to provide low-level support for the I/O functionality of the GR712RC.

The project started in February 2015 and completed in June 2016.

This paper describes the common DPU platform for ESA JUICE mission instruments and then continues with a discussion on the protocol software support for SpaceWire in use on the platform. It also describes tools and techniques used for unit testing and validating SpaceWire-based software in simulation and on target hardware.

II. DPU processor section

In the DPU architecture proposed the GR712RC LEON3-FT dual-core processor [2] from Cobham Gaisler is a natural choice to meet the requirements of CPU processing performance, memory architecture and I/O interfaces. It is beneficial from a power consumption, complexity and performance perspective using the GR712RC in this design since it provides all the identified I/O interfaces without the need to interface to additional components. With the clock-gating, I/O pin multiplexing and memory interface options, the GR712RC allows a power effective and modular design concept that is configurable to each instrument's specific needs. The DPU design memory configuration options, I/O interfaces and major component selections are summarized below.

- GR712RC 2 x LEON3-FT, 32KiB cache [2]
- Boot Memory [6]
  - 32KiB PROM (UT28F256LVQLE)
  - 2MiB MRAM (UT8MR2M8)
- Application memory [6]
  - 2MiB MRAM (UT8MR2M8)
  - 8MiB MRAM (UT8MR8M8)
- Working memory [6]
  - SRAM 4MiB BCH (UT8R1M39)

- SRAM 8MiB BCH (UT8R2M39)
- SRAM 16MiB BCH (UT8R4M39)
- SRAM 32MiB BCH (2xUT8R4M39)
- SDRAM 256MiB RS (UT8SDMQ64M48)

- 4 x SpaceWire (UT54LVDS031LV/E) [6]
- FPGA interface:
  - 32/16- or 8-bit I/O interface (buffered)
  - SpaceWire
- SPI, 6 x UARTs, GPIOs
- I2C, CAN, 1553B-MIL, Ethernet (free to use)

A block diagram of the DPU design is given in Fig. 1. Once the DPU is deployed in a final flight design the instrument manufacturer typically wants to configure and optimize schematics and optimize the layout for the instrument's specific needs. The analyses provided with the common DPU design may need to be adapted and refined for the flight board but serve as a strong starting point for the analyses reiteration.

A prototype system of the DPU design has been manufactured and is available in different configurations. The DPU prototype is a flight model instantiated on a 100mm x 100mm PCB carrying components of EM quality for all major components and logic.

The DPU prototype can be installed onto a commercial grade motherboard that provides connectors for all the listed interfaces, debugging capabilities, FPGA expansion slot, GR712RC switch matrix configuration, etc. The motherboard features multiple separate voltage rails, individually configurable voltage levels and voltage/current measuring circuitry which was used during hardware verification to characterize the power dissipation and to test the accuracy under +/-10% and 0% of the nominal voltage supply levels.

The photo in Fig. 2. illustrates one of the manufactured DPU prototypes. Centered is the GR712RC processor and to the right of it is an MRAM type application storage memory. Boot memory, working memory and SpaceWire transceivers are located on the opposite side. At top and bottom of the photo are the connectors for mounting the DPU prototype on its motherboard.

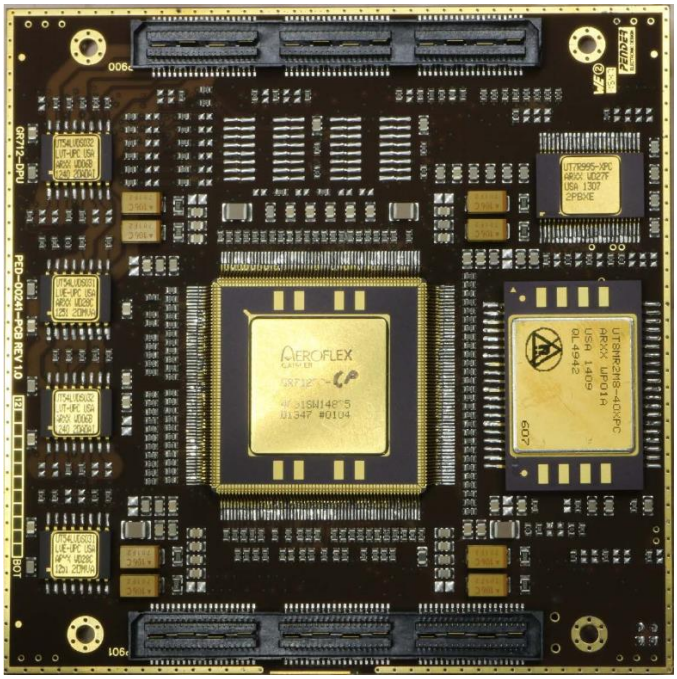


Fig. 2. DPU EM Prototype (back side)

III. Hardware Driver Software Library

The purpose of the Hardware Driver Software Library

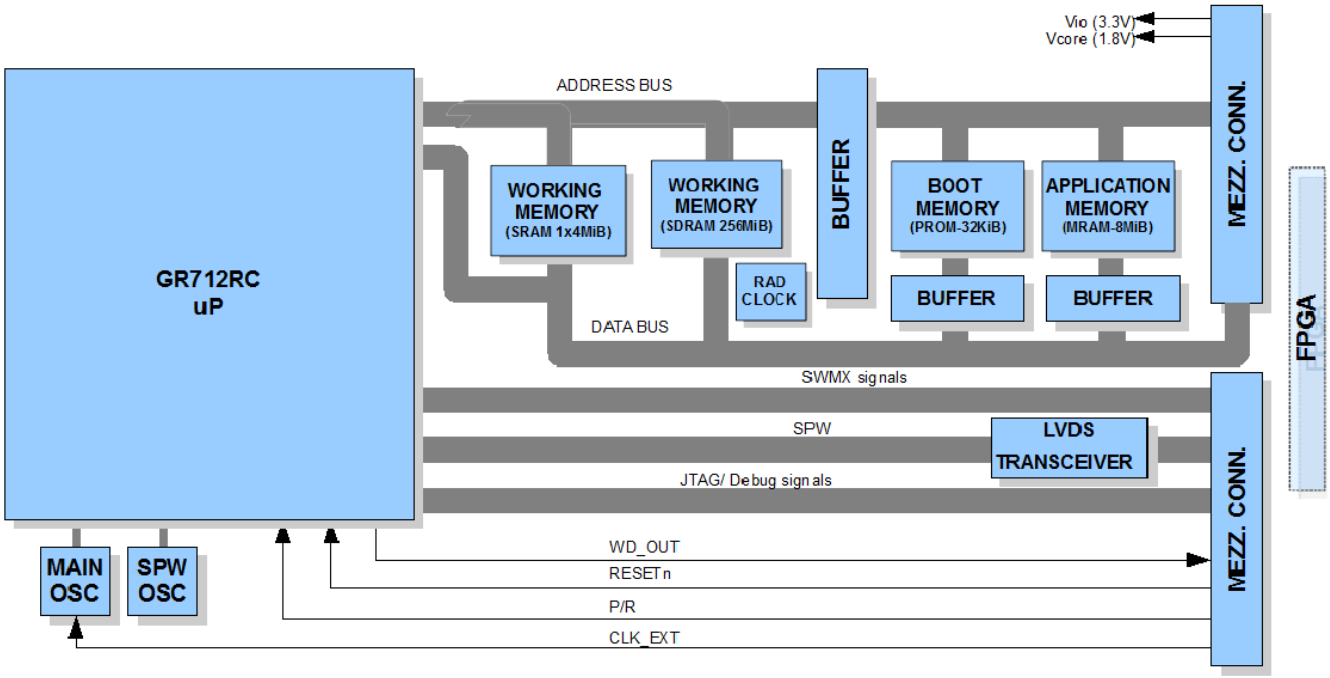


Fig. 1. DPU block diagram

(HDSW) is to provide drivers for operating a subset of the peripheral devices available on the GR712RC. Supported interfaces include SpaceWire, UART, SPI, GPIO, timers and more. This software is provided to the instrument manufacturers for incorporating in their instrument specific application software.

The drivers are compatible with RTEMS 4.10 [8]. However, all operating system services used by the driver library are called via an operating system abstraction layer, which in effect makes the driver library OS independent. For example, the DPU Boot SW uses the timer and SpaceWire drivers from HDSW, but with a custom OS backend adapted to the Boot SW run-time.

A user friendly Application Programming Interface (API) has been defined with the instrument application programmer in mind. All drivers operate in nonblocking mode with the option for the user to install interrupt handlers if required by the application. RAM buffers used by the drivers are allocated statically either by the user or by the driver. It is possible to operate the drivers without relying on dynamic memory allocation.

IV. DPU BOOT SW

DPU Boot SW is responsible for taking the DPU from system reset state to the execution of scientific instrument application software. The software consists of three parts which execute in sequence: P*rocessor Module Initialization*, S*tandby Mode* and A*pplication Loader*. The Boot SW implementation represents a tailoring of the ESA published requirement document *Flight Computer Initialization Sequence* [5]. This software is designed to execute on each of the JUICE instruments on startup.

The P*rocessor Module Initialization* sequence is responsible for configuring the GR712RC [2] hardware and perform self-tests on system resources. Any hardware functions which are not required for the operation of Boot SW are configured in a disabled mode. As each self-test progress, the result is written in a boot report which is later available via SpaceWire network service and to the instrument application. The early stages of the initialization sequence is implemented in SPARC assembly, and later when the RAM resource is made available, a C runtime environment is setup and used.

Standby Mode implements a PUS (Packet Utilization Standard) [9] terminal operating over SpaceWire. The PUS protocol has been tailored for low implementation complexity, high testability and for performance. Remote services are provided for managing on-board memory and to perform system specific operations such as protecting the MRAM resource. The memory service allows for managing the application storage memory, for example in-mission remote patching of an application image. RMAP is also available for low-level system access.

One SpaceWire interface at a time is used by *Standby Mode*. Dual redundant SpaceWire links are supported by a link reconfiguration manager implemented in software which can select among two interfaces. A scrubbing service is responsible for updating external RAM and caches memories.

SpaceWire Time Distribution Protocol (TDP) [10] is used to synchronize time with the on-board computer and all PUS telemetry packets are timestamped with the synchronized time.

The *Application Loader* is started when *Standby Mode* terminates. It can be triggered by a PUS TC command or a PUS TC timeout condition. Its purpose is to load, verify and start executing an instrument specific application software. A flexible application image format has been defined which allows for dividing the application software into individual sections where each section is protected with a CRC16 value. This allows for remote partial application patching using the PUS memory service.

V. SOFTWARE DESIGN METHODOLOGY

A rich set of configuration parameters are available for customizing the DPU Boot SW. This includes setting memory and SpaceWire properties and also allows for setting mission and timing properties. In particular, the full range of supported DPU configurations is covered by the software configuration parameters.

DPU Boot SW can be configured to execute on the GR712RC development board [11] in case the JUICE DPU hardware is not available.

A set of methodologies are being used to reach a high level of software determinism and to ease software testing:

- Boot SW uses its own custom run-time and thus does not depend on an operating system.
- Boot SW operates without interrupts and HDSW is not dependent on it.
- Boot SW has as default action to restart the system in the event of an unexpected trap.
- No dynamic memory allocation is used by Boot SW or the driver library.
- Third party source code is not used. The C standard library is not used.
- Hardware parameters are not probed by software.

VI. NETWORK PROTOCOL STACK

The SpaceWire network stack as used in the JUICE DPU Boot SW consists of the following components, from lower to higher protocol level:

- GRSPW2 (SpaceWire hardware interface)
- SpaceWire packet driver (*HDSW*)
- RMAP (GRSPW2)
- Time Distribution Protocol (Standby Mode)
- Link Reconfiguration Service (*Standby Mode*)
- CCSDS Packet Transfer Protocol (*Standby Mode*)
- Packet Utilization Standard (*Standby Mode*)

Careful design allows for network data buffers to be transferred up and down the stack without involving copying of data with the CPU. In general this is achieved by passing pointers between the layers after performing the necessary encapsulation/decapsulation and header validations. The protocol components are described in following sections.

VII. DPU SpaceWire hardware

The GR712RC provides up to 6 SpaceWire hardware interfaces (GRSPW2) where the first two features a hardware implemented RMAP target. GRSPW2 provides an interface between the GR712RC system bus and the SpaceWire network and provides DMA access to the DPU memories. The DPU schematics include the two first SpaceWire interfaces and two are optionally free to use. Up to two SpaceWire interfaces are used and controlled by the DPU Boot SW.

VIII. SpaceWire packet driver

The SpaceWire driver is part of the HDSW driver library. It provides a packet-based zero-copy interface, a link configuration interface and time-code interface. Memory footprint for the SpaceWire driver is less than 4 KiB, excluding packet buffers.

The driver *link interface* provides software control of how the GRSPW2 interfaces to the SpaceWire link. It allows for configuring SpaceWire clock frequency, node addresses, link state and receiving Time-codes. Status monitoring such as link state is also supported.

RMAP functionality is controlled by the link interface but the actual RMAP protocol is implemented in hardware.

The purpose of the driver *packet interface* is to provide a software API to the GRSPW2 DMA channels. Each SpaceWire packets is associated with a software structure describing the packet. These software structures can be linked together for the user to perform driver operations on multiple packets in one go.

Manipulation of GRSPW2 DMA descriptor tables is the main responsibility of the *packet interface*: when the driver is supplied with a list of packets, the descriptor table is updated accordingly with the new packets. One key point is that the driver never copies or interprets the packet payload data (*zero-copy*). It also means that the processor time consumed by the driver does not depend on payload sizes.

The main descriptor table manipulation operations available to the user are:

- grspw_dma_tx_send(): Schedule a list of packets for transmission.
- grspw_dma_tx_reclaim(): Reclaim packets which have previously been scheduled for transmission with grspw_dma_tx_send().
- grspw_dma_rx_prepare(): Provide driver with RX packet buffers for future DMA reception.
- grspw_dma_rx_recv(): Get received RX packet buffers back from the driver.

IX. Link Reconfiguration Service

Two separate SpaceWire interfaces in redundant configuration are available on the DPU. A link reconfiguration service is designed based on ESA requirements for the on-board network for the purpose of mitigating against sporadic and long-term network errors. The service runs periodically in Standby Mode operation to determine which of the two SpaceWire interfaces shall be selected as the primary interface.

A link quality algorithm qualifies the nominal and redundant interface by monitoring them for link state changes and for link errors. The qualification outcome is used to determine which of the SpaceWire interfaces to select as the primary interface to operate on. In *Standby Mode* operation, the primary interface is used for sending TM reports. Valid TC commands are always processed, independent on which interface they arrive on. The primary interface is also used for the SpaceWire Time Distribution Protocol (TDP).

Link reconfiguration is optional as the DPU can be configured with only one SpaceWire interface (dual-redundant DPU configuration).

X. CCSDS Packet Transfer Protocol

CCSDS packet transfer protocol [12] is a protocol for transferring CCSDS packets across a SpaceWire network. The protocol does not guarantee packet delivery or packet order, and there is no transfer confirmation and no quality-of-service provided. A simple encapsulation / decapsulation scheme is defined [12], with two main services:

- A transmission service is called from the PUS layer and encapsulates a CCSDS packet into a SpaceWire packet for processing by the SpaceWire packet driver.
- A reception service gets a SpaceWire packet from the SpaceWire packet driver layer. It decapsulates it and eventually passes a CCSDS Space Packet up to the PUS layer. Target logical address and protocol identifier fields are checked according to configuration parameters.

Since the protocol CCSDS Packet Transfer Protocol field length is known and fixed, the Standby Mode implementation does not perform any copying in this layer. After validation, the data buffer is forwarded with an adjusted pointer value.

XI. Packet Utilization Standard

The Packet Utilization Standard (PUS) [9] is used for end-to-end transport of telemetry and telecommand data between user applications on the ground and application processes on-board the DPU. Specifically, PUS defines application-level interfaces between ground and space.

PUS relies to a large degree on mission tailoring, both with regard to the set of supported services and with regard to packet structure layouts. In the case of *DPU Boot SW*, the services tailoring is defined by the *Flight Computer Initialization Sequence* [5] while packet structure tailoring was set by ESA and Cobham Gaisler as part of the project specification. Driving factors for the tailoring has been to reduce implementation complexity and achieve high testability and run-time performance.

Supported services include:

- Telecommand verification (PUS service 1)
- Housekeeping (PUS service 3)
- Event reporting (PUS service 5)
- Memory management (PUS service 6)
- Function management (PUS service 8)
- Connection test (PUS service 17)

XII. SpaceWire Time Distribution Protocol

The DPU implements a SpaceWire Time Distribution Protocol (TDP) [10] client for synchronizing its local time with the on-board time from an on-board TDP master.

Time information is transferred by an RMAP write command carrying a CCSDS Time Code. The synchronization event is signaled by means of transferring a SpaceWire time control code (Time-Code) which activates the corresponding CCSDS Time Code.

Both RMAP and SpaceWire time control codes are supported in hardware by the GR712RC. However, the TDP protocol itself is not directly supported. The solution was to make use of the hardware support and implement the time synchronization/activation in software. When a SpaceWire time control code arrives, software compares it with the last CCSDS Time Code received over RMAP to qualify it. To calculate time-offsets for the local time, the software implementation uses an event time-stamping functionality available in the processor.

One time synchronization per second is accepted by the *Standby Mode* as demanded by the mission requirements. Each PUS telemetry packet sent by the DPU contains the local time.

XIII. Unit testing

The SpaceWire software unit test effort aims to provide confidence that the various software components comply with their specification on a per-function level. These tests are designed to execute on both the target prototype hardware and in the TSIM2 SPARC/LEON3 simulator [13]. A unit test framework has been developed which allows script-based execution and logging of the tests.

Software units are tested in isolation by using mock routines at the borders of the tested unit. A custom assert-like function library is implemented in the unit test framework and used to describe and verify the expected unit behavior and invariants.

Execution times for the SpaceWire driver are recorded by the unit tests when running on prototype hardware. This is achieved by the test framework sampling a timer. Measuring time samples recorded while executing the functions under test with various function parameters are used to provide realistic bounds on execution time.

TSIM2 [13] is integrated into the test framework to automatically extract code coverage reports. Instruction level code coverage for the JUICE SpaceWire software is 98 %. For software branches not covered by unit tests, the code has been manually analyzed and put into context with corresponding validation test cases to provide evidence of full coverage.

XIV. Validation

A validation framework for the DPU software has also been developed. It is used to validate the software requirements and exercise the full DPU software, including the SpaceWire network stack.

In the validation setup, DPU Boot SW runs on the DPU prototype board connected with two SpaceWire interfaces to a GRESB SpaceWire/Ethernet bridge [14]. The GRESB is connected to a workstation PC. The PC executes a validation test suite which sends PUS telecommands over the SpaceWire network via the GRESB to the DPU (PUS terminal). Side-effects of the telecommands are verified by investigating PUS TM reports. The side-effects can also be timeouts, restarts, link reconfigurations or expected TM omissions. The validation test setup is illustrated in Fig. 3.

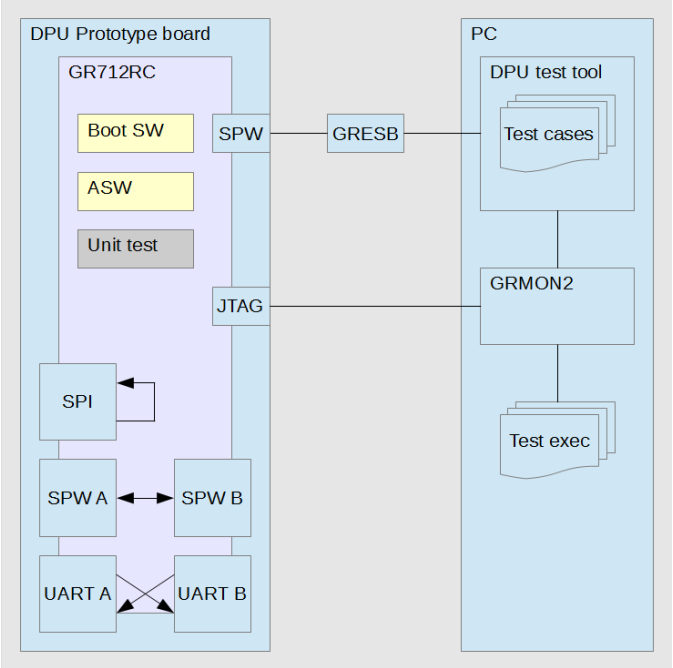


Fig. 3. DPU validation setup

At the heart of the validation framework is a Tcl [15] library developed for communicating with the GRESB and simplifying common operations such as matching TM responses, performing multiple TC uploads and system restart. This gives a streamlined path from specification of validation test cases, to test implementation and execution.

A large part of the validation framework consists of the PUS TC generator and TM validator. Functionality is also available for generating TDP traffic, injecting erroneous packets on different network layers, and for controlling the SpaceWire link state. Link state control is used to validate operation of the Boot SW link reconfiguration service.

A set of sample application software images have been prepared to validate the system state at handover to application software.

To facilitate visibility of temporal software behavior, the DPU Boot SW has a built-in execution time monitoring per task which can be used to bound effective execution time. Time measurements are reported via a housekeeping report service monitored by the validation framework running on the PC.

A separate driver validation software, running on RTEMS 4.10 [8], is also available. It exercises all the different *HDSW* drivers in parallel on the DPU prototype hardware.

XV. Compliance to Software Engineering Standards

One of the baseline requirements for the project was to develop the DPU Boot SW and Hardware Driver Software Library in accordance with ESA software engineering and product assurance standards [3], [4]. A set of deliverables, document and source code, was agreed upon and the software criticality was set to category B.

A specification phase was carried out which included preparation of software requirement specifications, interface control documents, unit test plans and validation test plans, including test case specifications and expected use-cases. This phase also included development of software design documents.

The software implementation phase followed in combination with development of the unit test framework and implementation of unit tests. Hardware and system simulators were integrated in the unit test loop as well as functionality for performing automated code coverage analysis (simulation) and execution time measurements (hardware).

As the software implementations matured, the DPU validation framework was developed. The previously specified validation test cases were implemented using the validation framework and the corresponding equipment such as the GRESB SpaceWire/Ethernet bridge [14].

The software validation phase was matched with arrival of the prototype boards which allowed for running the software on a representative hardware platform.

When work with the specifications, implementations, testing, validation and documentation was completed, the project was closed after a successful review with ESA.

XVI. Conclusion

A flight quality SpaceWire driver, together with a SpaceWire network protocol stack has been implemented and is available for the GR712RC LEON3-FT [2].

It includes software support for raw SpaceWire packets, Time Distribution Protocol (TDP), a link reconfiguration service, the CCSDS Packet Transfer Protocol and Packet Utilization Standard (PUS). The software is compliant with ESA ECSS standards for Space software engineering [3], [4]. In addition, the DPU platform provides support for RMAP, SpaceWire time-control codes and the SpaceWire data link layer protocol.

By using the presented SpaceWire software, a common set of services can be implemented by maintaining only one piece of software. Using already validated boot, low-level driver software and the automated test framework reduces the effort of the instrument software development and demonstrates a viable approach of software design and testing accepted by the agency.

The software is available as a standard software to the JUICE instrument manufacturers for use in the JUICE mission. Cobham Gaisler can also license the software to customers for use in other missions.

The DPU flight hardware platform design is delivered as design files (schematics, layout, BOM). It comes with a set of quality documents, for example a Failure Mode and Effects Analysis (FMEA) and a Radiation Analysis.

Prototype DPU boards have been manufactured. These systems have been used for verification of the design and for development of the DPU software.

Although the instruments in the ESA JUICE mission demands a wide range of memory configurations, performances, interface peripherals and software support, the DPU developed within the activity can house all major configurations at performance. Thus it provides the possibility to use it as a common DPU platform the among instruments.

The gain of using a common DPU hardware and software platform is not only to the user's but is also an advantage for ESA and the satellite prime contractor as the overall satellite design is harmonized. The instrument development process can be shortened by using an already defined DPU and software package. As the DPU is modular is can be reused in future missions with similar or less demanding environmental constraints.

Acknowledgment

The development of the common DPU platform hardware and software was funded by the European Space Agency as part of the activity "DPU for JUICE Instruments", contract 4000113396/15/NL/BW.

References

[1] European Space Agency. ESA Science & Technology: "JUICE – Jupiter Icy Moons Explorer" [Online]. Available: http://sci.esa.int/juice.

[2] Cobham Gaisler. "GR712RC Dual-Core LEON3FT SPARC V8 Processor" [Online]. Available: http://www.gaisler.com/index.php/products/components/gr712rc .

[3] European Space Agency, "Space Engineering – Software. ECSS-E-ST-40C". March 2009.

[4] European Space Agency, "Space Product Assurance – Software Product Assurance. ECSS-Q-ST-80C". March 2009.

[5] European Space Agency, "Flight Computer Initialisation Sequence. TEC-SWS/10-373/FT". October 2014

[6] Cobham Microelectronic Solutions, "HiRel Memories" [Online]. Available: http://ams.aeroflex.com/pagesproduct/prods-hirel-mems.cfm.

[7] Cobham Microelectronic Solutions, "HiRel LVDS Family" [Online]. Available: http://ams.aeroflex.com/pagesproduct/prods-hirel-lvds.cfm.

[8] Cobham Gaisler, "RTEMS LEON/ERC32 Cross-Compiler System" [Online]. Available: http://www.gaisler.com/index.php/downloads/compilers.

[9] European Space Agency, "Space Engineering – Ground Systems and Operations – Telemetry and Telecommand Packet Utilization. ECSS-E-ST-70-41A". January 2003.

[10] European Space Agency, "High Accuracy Time Synchronization over SpaceWire Networks – Time Synchronization Protocol. SPWCUC-REP-0003, Version 1.1". September 2012.

[11] Cobham Gaisler, "GR712RC Development Board – User Manual" [Online]. Available: http://www.gaisler.com/.

[12] European Space Agency, “Space Engineering – SpaceWire – CCSDS packet transfer protocol. ECSS-E-ST-50-53C”. February 2010.

[13] Cobham Gaisler, “TSIM2 ERC32/LEON simulator” [Online]. Available: http://www.gaisler.com/index.php/products/simulators/tsim.

[14] Cobham Gaisler, “GRESB SpaceWire/Ethernet Bridge with routing capabilities” [Online]. Available: http://www.gaisler.com/index.php/products/systems/gresb.

[15] Tcl Developer Xchange, “Tcl Developer Xchange” [Online]. Available: http://www.tcl.tk/.

Towards SpaceWire-2: Space Robotics Needs

SpaceWire missions and applications, Long Paper

Olivier Notebaert❖ (Author)

❖ Spacecraft On-Board Data Processing Expert
Airbus Defence and Space SAS
Toulouse, France
Olivier.notebaert@airbus.com

Giuseppe Montano[1], Thierry Planche[2], Clément Pruvost[2], Franck Wartel[2], Andreas Schüttauf[3], Hans-Jürgen Herpel[4], Christophe Honvault[5], David Jameux[5]
(Co-Authors)

[1/]Airbus Defence and Space Limited, Stevenage, UK
[2/]Airbus Defence and Space SAS, Toulouse, France
[3/]Airbus Defence and Space GmbH, Bremen, Germany
[4/]Airbus Defence and Space GmbH, Friedrichshafen, Germany
[5/]European Space Agency ESTEC, Noordwijk, the Netherlands

***Abstract*— A number of evolutions of the SpaceWire standard [1] and usage recommendations have been developed over the past decade. As a result, the original standard is slowly evolving towards a so-called *SpaceWire-2* (SpW-2) standard family, covering legacy SpaceWire, SpW-Rev1 and SpaceFibre [2],[3].**

Besides other use cases that are mainly covered through on-going actions structured through the SAVOIR-UNION working group [4], the requirements of the future on-board network systems should also cover the needs of space robotics which is a rapidly evolving engineering discipline and one of the potential space applications of SpaceWire-2.

This paper presents a number of robotics-specific needs that should be taken in consideration for the definition of SpW-2, and investigates the capabilities provided by the SpaceWire standard and evolutions (SpW rev1 and mainly SpaceFibre).

***Index Terms*—SpaceWire, SpaceFibre, real-time networks, SAVOIR, robotic systems control, GNC, vision based navigation, on-board data processing, on-board software, space exploration, planetary exploration, in-orbit servicing, orbital systems, robotic arm, planetary rovers.**

I. INTRODUCTION

A number of evolutions of the SpaceWire standard [1], usage recommendations or communication protocols have been developed over the past decade, including for instance the SpW-R [5][6], SpW-D [7][8], SpW-RT [9], SpW-NDCP[10], or N-MaSS [11][12]. A SpaceWire standard revision is also expected to be released soon [13] and the SpaceFibre standard is currently being drafted [2] to provide higher performance and quality of service [14]. As a result, the original SpaceWire standard is evolving towards a so-called "SpaceWire-2" (SpW-2) standard family, which will cover an integrated solution to all the issues addressed by the evolutions mentioned above [3].

At the 6th International SpaceWire Conference in 2014, it was proposed that the requirements baseline for SpaceWire-2 should be assessed first by gathering the needs from various heterogeneous space applications, then by identifying key properties and creating an evaluation system (TABLE I.) to be used by the community that defines the standard [15].

TABLE I. PROPOSED EVALUATION SYSTEM IN [15]

METRICS			EVALUATION					
Domain	Use cases:		Satelites Platforms		Science missions Payloads		Launchers	
	Inputs projects (for lessons learned) or studies (on future missions):		MISSION, SpW-D, AOCS SpW, N-Mass,...		Bepi Columbo, Solo, N-Mass		Ariane-5 Avionique-X	
	Type of applications:		Data handling, Spacecraft AOCS, FDIR		Payload Data handling and processing, instruments control		Guidance, navigation and control / Flight telemetry	
	Properties	Units	Value[1]	Relevance[2]	Value[1]	Relevance[2]	Value[1]	Relevance[2]
Electrical	Network consumption	mWatt/device	50	2	< 50	3	100	1
	ElectroMagnetic	High/Medium/Low	Medium	3	Medium	3	High	3
	Bus Error rate	Typical value	1,E-12	3	1,E-12	2	1,E-14	3
	Media	Copper/Fibre/Both	Both	1	Both	2	Copper	3
	Clock transmission	Yes/No	Yes	2	No	1	Yes	3
	Architecture	P2P/Bus/Network/Ring	Network	2	Network	2	Bus or Network	2
Mechanical	Connectors type	fixed/few variants/free	Fixed	1	Fixed	1	Few V.	2
	Connectors weight	g	10	2	10	3	10	1
	Connector Vibration resilience	Low/Medium/High	Medium	2	Medium	2	High	3
	Tooling for mating	Yes/No	Yes	2	Yes	2	Yes	2
	Harness Mass	g/m	< 80	2	< 80	3	80	1
Functional	Gross data rate	Gigabit per second	1 - 5	1	1 - 10	1	0,1 - 1	1
	Minimum Net data rate	MegaByte per second	> 100	3	> 1000	3	> 20	3
	Retry	Yes/No/Optionnal	No	2	Yes	1	No	3
	Time-distribution	Yes/No/Optionnal	Yes	3	Optionnal	2	Yes	3
	Flow Control	Yes/No/Optionnal	Yes	2	Yes	3	Yes	3
	High Priority message	Yes/No/Optionnal	Optionnal	2	Optionnal	2	Yes	3
	Error detection	Yes/No	Yes	3	Yes	2	Yes	3
Implementation Constraints	Master IP size	Eq ASIC Gate, KiloBytes	20	1	20	1	20	1
	Slave IP size	Eq ASIC Gate, KiloBytes	10	3	10	3	5	3
	Stand Alone component	Yes/No	Yes	3	Yes	3	Yes	3
System features	Bridge to other buses	1553/Can/Other	1553/Can	1	SpW	3	1553	2
	Link length	m	10	2	10	2	60	2
	Failsafe	Yes/No	Yes	2	Yes	1	Yes	3
	Redundancy management	Yes/No	Yes	2	Yes	2	Yes	2
	Robustness against babbling idiot	Yes/No	Yes	3	Yes	3	Yes	3

1. **Value**: Typical value or range for the considered property w.r.t. the considered application domain
2. **Relevance**: Property relevance for a given mission (3: High, 2: Medium, 1: Low, 0: N/A)

Such work has not been formally done following this methodology in a cooperative mode as proposed. However, the common European vision of the future satellite platform and payload data handling architecture has been structured through the SAVOIR initiative which has improved visibility on common needs [16]. Quite recently, the work focusing on future user needs in on-board network has been finally launched with the SAVOIR-UNION working group [17].

All these actions however do not specifically focus on future space robotics which is a rapidly evolving engineering discipline and one of the potential space applications of SpaceWire-2. In fact, Airbus Defence and Space is currently investigating the use of SpaceWire in a number of planetary exploration robotics and orbital robotics applications: specific requirements and issues with the current version of the standard are being identified within a number of past and on-going studies and demonstrators.

A number of robotics-specific needs should be taken in consideration for the definition of SpW-2, covering aspects such as mechanical design of cables and connectors, data handling architecture, FDIR and network management. In front of this requirements envelope, an analysis of the capabilities provided by the current state of the SpaceWire standard system (SpaceWire, SpW rev1 and SpaceFibre) identifies potential remaining gaps that the future SpaceWire-2 should fulfil to fit the future robotics missions' needs.

II. Future robotics missions and use cases

We can easily support the vision that the growing share of automation in many complex ground applications (as human assistants first and then in full autonomy) shall logically be derived in space sooner or later. Currently, the extremely fast development of robotics applications on ground together with the progress in sensors and effectors critical technologies, mechatronics, microelectronics, data processing and software enables the development of innovative concepts that will increase science return with in-situ robotics. Compared with the past decades, we can expect a drastic increase of the use of robotics systems in future space mission. This applies especially for planetary exploration and in-orbit operations.

Planetary Observation

Future planetary exploration missions planned by ESA following ExoMars 2020 are for instance:

- Phobos Sample Return (PHOOTPRINT): a sample-return mission to the Mars moon Phobos proposed for launch in 2024. This mission will require a robotic arm equipped with sampling tools (Fig. 1)
- Mars Sample Return (MSR) will include a robotic arm dealing with the return of the sample cache from a mobile platform (Fig. 2)

Several concepts are being investigated to return samples from asteroids (e.g. ESA Asteroid sample return, ESA Marco-Polo), or planetary moons such as the Moon or Phobos (e.g. ESA PhootPrint, Phobos Sample Return). All these concepts involve a single lander platform with a robotic manipulator tasked to retrieve a sample from the surface. A variety of robotics tools will have to be connected to, and used by, the robots involved in the mission. The low gravity, vacuum and dust environment will be critical in the design of such end-effectors and their interfaces.

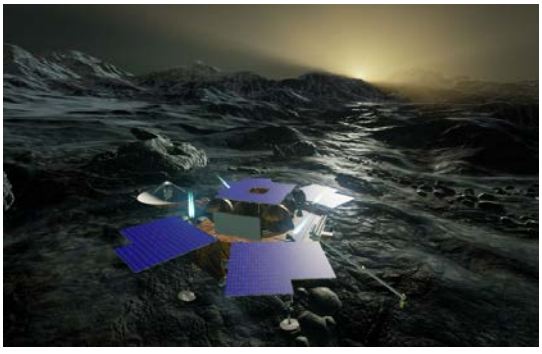


Fig. 1 - Phobos Sample Return Lander Concept with the sampling arm extended

Rovers and robotic arms/manipulators are used in most of the missions being considered. A robotic manipulator can be used to facilitate the placement of payload heads onto the surface (e.g. Raman/LIBS spectrometer, microscope) or to deploy specific payload elements (e.g. seismometer). When numerous operations need to be performed by a single manipulator, a tool exchange system can be used to swap end-effector to carry out various tasks such as trenching, drilling or in-situ imaging and analysis.

In terms of rovers, the current MSR concept relies on a main lander element that includes a Mars Ascent Vehicle (MAV) for the return of the sample, a Sample Fetching Rover (SFR) and a sample cache, deposited earlier on the surface of Mars by a caching mission. The SFR is deployed from the MSR lander or a secondary platform and is tasked to recover a passive sample cache by means of a robotic manipulator. The rover then returns to the MSR lander and the sample cache is transferred to the MAV (by the rover manipulator or a separate lander arm).

Tool exchange as well as the variety and complexity of the various vehicles mission modes poses several issues on the data handling system as in particular the scalability and the capability to partly, dynamically and autonomously reconfigure communication networks with change of functional interfaces.

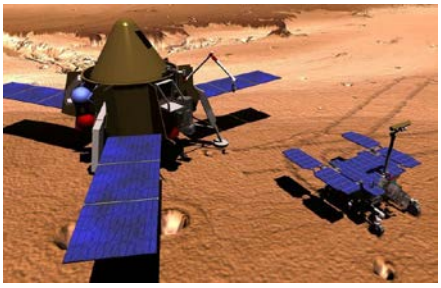


Fig. 2 - Mars Sample Return Lander Concept with the arm dealing with the return of the sample cache from a mobile platform

In-Orbit Operations

Closer to Earth, the need for robotics systems for autonomous manoeuvres and manipulation will increase. Future missions put a particular focus on debris removal but also on in-orbit spacecraft servicing (Fig. 3).

Missions in preparation are for instance:

- e-Deorbit, for making rendezvous and docking with the old and large ENVISAT spacecraft target and steering down for a controlled burn-up in the atmosphere (Fig. 4, [18][19]);
- more generally the whole Clean Space initiative of ESA for improving technologies enabling to clean the thousands of accumulated space debris in GEO and LEO orbits [20];
- the Airbus Defence & Space Space-Tug concept [21], containing fuel and powered by electrical propulsion, staying in a parking orbit for servicing spacecraft for various types of missions such as spacecraft tugging from Low Earth Orbit to GTO or cis-lunar orbit, in-orbit spacecraft repair, upgrade, refuel or reboost, debris tugging to controlled re-entry etc…

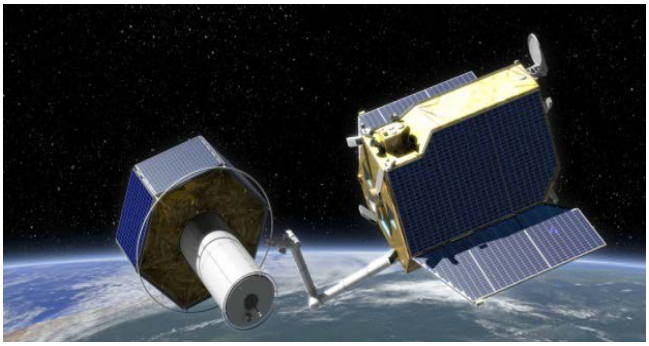


Fig. 3 – DEOS concept for In-orbit satellite servicing [22]

Fig. 4 - e-Deorbit concept for ENVISAT capture ([19])

In all these missions, the capabilities to navigate autonomously as well as in-situ manipulations are necessary. This will require robotic arms with an adapted variety of tools, visual and other local sensors and a real-time on-board processing capability. So the identified critical technologies that need to be further developed or improved are Vision Based Navigation (VBN) and on-board high performance processing for in-situ images analysis and autonomous real-time control. In all these systems, sensors and control data will require to be transported from various points through datalinks that will induce specific requirements for real-time performance and reliability.

In other words, to achieve future robotics missions, spacecraft bodies will require better brains and be completed by eyes, articulated arms, hands with agile fingers possibly capable of touch feeling, and few other sensors and tools. Of course, a network of nerves (data links) will be needed to carry data between all these.

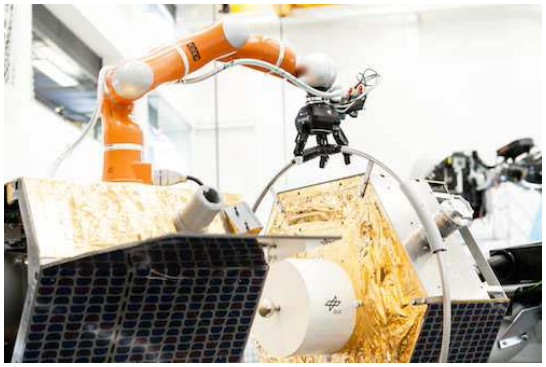


Fig. 5 – Spacecraft gripping demonstration, source DLR [23]

III. Physical layer

Mechanical constraints may apply to robotics systems. An analysis of these constraints w.r.t. to the current SpaceWire standard is provided below with some recommendations as input to SpaceWire-2 definition.

Cables

The SpaceWire cable consists of four 100 Ohm differential twisted pairs for each link, each with a shield and jacket; these are encased around a central filler and enclosed with an overall shield and protective jacket. This configuration as shown below (Fig. 6) presents a very robust cable assembly with good EM shielding and high signal performance. However, the robustness of this solution is paid in terms of non-negligible harness mass and cost brought to the data-handling hardware.

The standard SpaceWire cable has a dynamic bend radius of 60 mm, which is suitable to most of satellite applications but can easily become a limiting factor with robotics, which are usually made of modular and movable parts. In fact, in a recent feasibility study made by Airbus Defence and Space in UK about the implementation of a full SpaceWire-based control system for the LARAD arm, it was found that the standard SpaceWire cable would impair the degrees of freedom of the arm. Detailed analysis revealed that a maximum bending radius of 15 mm must be met in order to preserve the mobility of the parts. Several non-standard cable products are available for instance from Axon that bring better bending characteristics with less weight (TABLE II.).

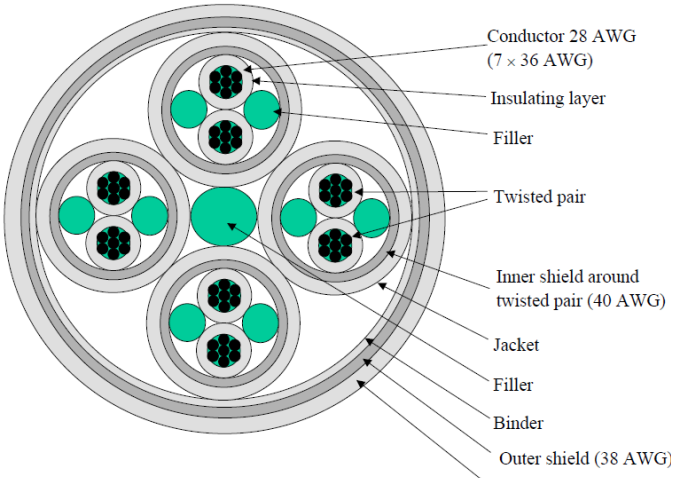


Fig. 6 – Standard SpaceWire cable [1]

Connectors

Miniaturization is extremely important for many robotic applications, especially in space exploration missions for which every gram, cm3 and Watt counts. The standard Sub-D connector is not really miniaturized. SpaceWire specifies the use of 9 pin micro-D connectors: despite of being quite smaller than the Sub-D connector, these still require a large amount of both space on the board for the socket, as well as room behind the connector to properly terminate all the shields into the connector. The size of the micro-D is 7.57 x 19.69 x 10.57 mm.

Axon proposes a new alternative, the nano-D (size: 3.18 x 9.53 x 5.33 mm) (Fig. 7), which may bring benefits to robotics. Such a miniaturized connector should be proposed at least as an option in SpW-2.0.

Aging and damage with repetitive mating and un-mating is generally not a problem is satellites but may be a problem with robotics (e.g. connection/disconnection of robotic modules, robotic arms with exchangeable end-effector). Connectors shall also be preferably designed to resist to many mate/un-mate cycles.

Dust is not a problem for orbital applications but it is a problem for planetary exploration. Mars, for instance, has dust and strong winds, hence exposed connectors have to be designed to operate under these hostile conditions. Industrial solutions have been proposed (e.g. dust-tolerant connectors by honeybee, Fig. 8) which may be taken in consideration. Unfortunately, these solutions are far from being miniaturised!

TABLE II. CABLE OPTIONS MAIN CHARACTERISTICS
SOURCE:www.axon-cable.com [25]

	SpaceWire ESCC 3902.003.01	Low Mass SpaceWire ESCC 3902.004.01	Ultra Low Mass Coax Link with overall shield	Ultra Low Mass Coax Link without overall shield
Mass (g/m)	80 max.	42 max.	32.5 max.	30 max.
Overall Ø (mm)	7 max.	6.5 max.	4.5 max.	4.2 max.
Static bend radius	45	25	10	6
Dynamic bend radius	60	30	20	15
Impedance (Ω)	100 ±6	100 ±6	2x50 ±2	2x50 ±2
Capacitance (pF) - intra pair - inter pair	 < 50 < 90	 < 50 < 90	 < 48 < 97	 < 48 < 97
Resistance DC (Ω/m)	0.23	0.23	0.90	0.90
Intra pair skew (ps/m)	< 80	< 50	< 20	< 20
Inter pair skew (ps/m)	< 130	< 100	< 20	< 20
α (dB/m) @1 GHz Cable length (for -6 dB atten.)	-1.5 4.5 m max.*	-1.4 4.6 m max.*	-2.6 2.3 m max.*	-2.6 2.3 m max.*
Return Loss (dB) up to 2 GHz	-9 max.	-9 max.	-20 max.	-20 max.

*= for a 400 Mb/s data rate (1 GHz)

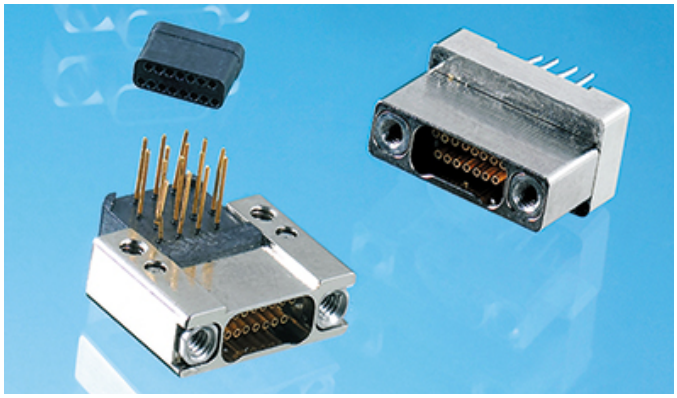


Fig. 7 – AXON Nano-D connectors
SOURCE: www.axon-cable.com [26]

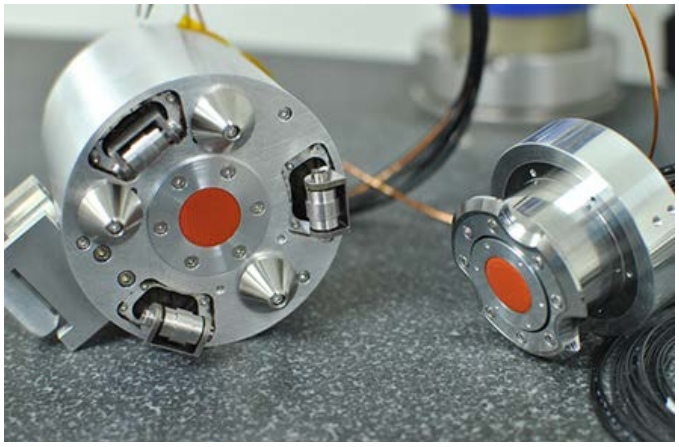


Fig. 8 – Honeybee robotics dust tolerant connector. The red membranes cover the electrical contacts before mating
SOURCE: www.honeybeerobotics.com [27]

IV. DATA HANDLING SYSTEM ARCHITECTURE

There are two main drivers that are viewed as specific to robotics needs in terms of data network architecture.

On the one hand, to carry on the analogy with human bodies above in this paper, a data-link network shall be the nerves linking the brain with eyes, arms, hands, fingers and all other potential sensors. Many peripheral systems inducing various network topologies shall be reliably linked together and with the brain (on-board processing device); this all depends on the type of specialized action the robot is made for, and it also may be moving and/or disconnected and reconnected w.r.t. each other. Flexibility of the network topology, modularity and easy in-flight re-configurability shall therefore be driving requirements for SpaceWire-2 within the picture of a full spacecraft data network in which a SpW/SpF based backbone may connect to a diversity of other sensor-buses such as CAN, SPI, I2C, MIL-STD-1553, PowerLink, etc…

On the other hand, beside the flexible topology issue, the autonomy is the other main issue in space robotics applications. This means autonomous real time control for critical operations which cannot be tele-operated by humans due to the space situation: recognizing targets, navigating for rendezvous, gripping, latching, clamping, screwing, etc… Indeed, except for manned flight that, in few cases, may allow for direct and interactive robot-human operations as, for instance, on the International Space Station, the signal delay itself between an Earth-based control center and the spacecraft is too long to allow for remote operations. Typical robotic systems applications imply hard-real-time closed loops with frequencies in the range of 100 Hz, which is sensibly above the classical spacecraft Attitude and Orbit Control requirements (typically 10 to 20 Hz). Finally, emergency action in case of failure or hazardous situation is also to be done autonomously: this is treated below in the section V on *FDIR and network management*).

TOPOLOGY

Within a robotic system, the topology of the data network, linking the controlling brain to all sensors and actuators, is

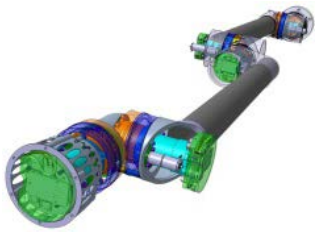


Fig. 9 – LARAD robotic arm

Fig. 10 – Bridget on the Mars yard in Stevenage

highly dependent on the application. For example, the Lightweight Advanced Robotic Arm Demonstrator (LARAD) robotic arm (Fig. 9) will have a *Daisy Chain* shape for the data network while a rover vehicle for planetary exploration like the Bridget demonstrator (Fig. 10) will have Tree like network, i.e. with multiple branches. Moreover, robotic arms may be mounted on larger robots with or without use of a single or several independent networks. Each project needs therefore to explore the space of architectural solutions, this being made easier thanks to networks modeling and analysis tools such as MOST [32], in particular to pre-check real time performance provided by the possible topologies.

In the case of the Lightweight Advanced Robotic Arm Demonstrator (LARAD), several topology options have been explored in the frame of the development of a control system architecture fully based on the SpaceWire [28]. The classical star topology, the bus chain and several variants of ring topologies have been traded-off to link the LARAD On-board Computer (OBC) with the Joint Electronics (JE) units and the End Effector (EE). The "*Ring with Interleaved Connections* topology (Fig. 11) is finally the preferred option with respect to harness mass, reliability and estimated performance of the network. Complementary analysis has also been performed on these topologies to determine the network data latencies typical and worst cases for the nominal case and with a disconnected link (Fig. 12). This analysis confirms this topology as the best option (detailed results are presented during this SpaceWire conference [29]).

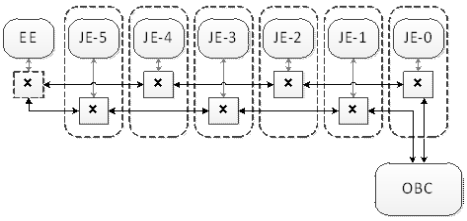


Fig. 11 – LARAD SpaceWire Network, Ring Topology with Interleaved Connections [28]

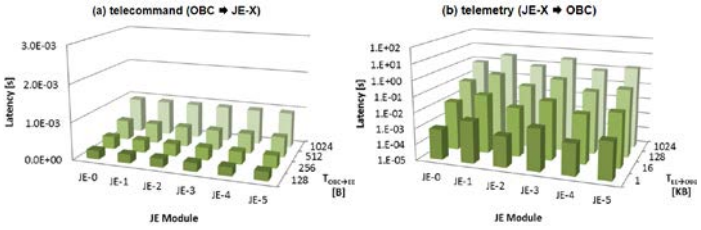


Fig. 12 – Worst-case SpaceWire latencies for the telecommand (a) and telemetry (b) data flows in the interleaved ring network topologies with failures at 10Mbps [29]

V. FDIR AND NETWORK MANAGEMENT

The on-board network topology of robotic systems often changes during operation due to connection/disconnection to other robots or spacecraft and, in case of arms, due to end-effectors exchange. Moreover, a common driver for most robotic missions is that the effect of faults may result in physical damages of part of the system thus potentially jeopardizing the mission. And when human beings (e.g. astronauts) or human vital assets (e.g. inhabited space vehicle) are involved in the operations, it becomes safety critical. In this context, a first level FDIR function that is reactive, autonomous and integrated within the network management function may be extremely beneficial.

The SpW-FDIR protocol ([11],[12]), later to be integrated into a wider Network Management Service Suite (N-MaSS), provides a solution for simplifying network management and autonomous network level fault detection, isolation and recovery. It has been defined for SpaceWire networks and provides additional functions that could be standardized to be included in SpaceWire-2, i.e. SpaceWire/SpaceFibre networks. SpW-FDIR manages network topology and configuration, node identity and configurations. It autonomously maintains the connectivity and performance of data handling networks in the presence of failures.

The SpW-FDIR components as part of the N-MaSS architecture are illustrated below (Fig. 13). Main features are:

- SpW-FDIR is scalable and can be applied for networks up to 256 nodes;
- SpW-FDIR is designed for extremely fast response
- Protocol overhead is minimal
- SpW-FDIR requires a FDIR Handler function within each node and switch

SpW-FDIR sit in the Network Layer of SpaceWire. At this level, it is well-positioned to interact with 'Device Discovery' and 'Resources Management' functions (Fig. 14); this is highly beneficial for systems that have reconfigurable topologies (i.e. modular, reconfigurable robots).

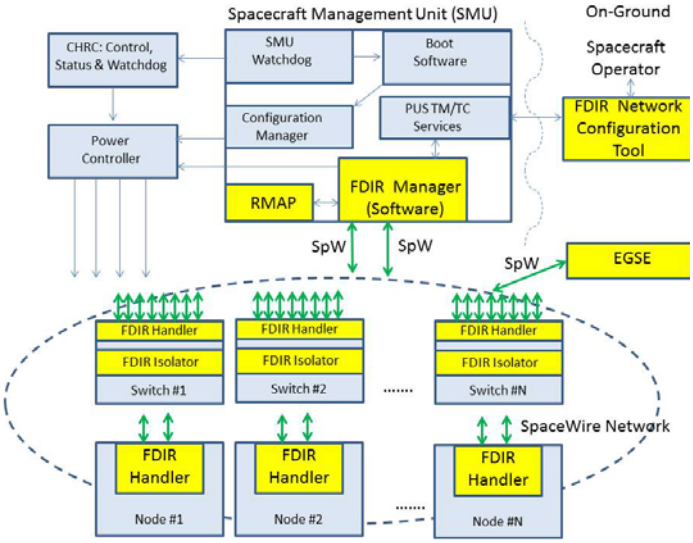


Fig. 13 – SpW-FDIR components as part of N-MaSS architecture [12]

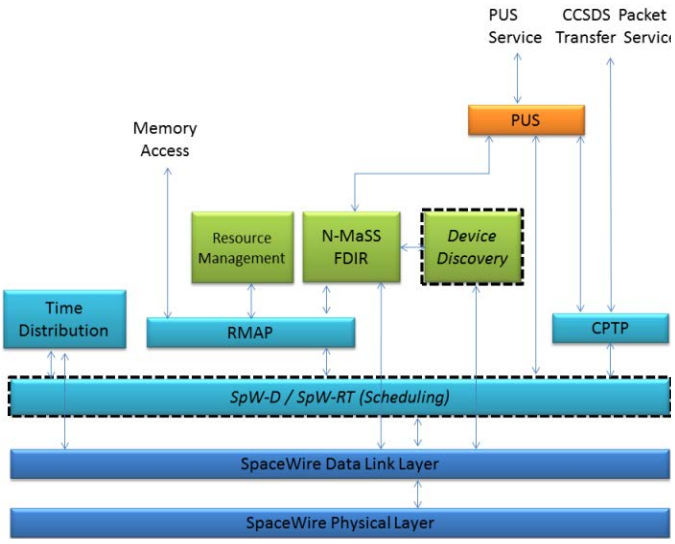


Fig. 14 – Context and N-MaSS position in the SpaceWire protocol stack

VI. SpaceWire / SpaceFibre Functional Properties

The current SpaceWire ECSS standard [1] and SpaceFibre draft specification [2] should basically allow to efficiently implementing most of the space robotics data links. However, a few weaknesses w.r.t. high level user needs such as, for instance, some physical layer properties as discussed in section III above, may require improvements. This could either be introduced as standard evolutions (e.g. the addition of a SpaceWire/SpaceFibre networks management protocol), through options in the future SpaceWire-2, or treated at project level by classical waivers mechanism. Some typical functional features for the future SpaceWire/SpaceFibre networks are now discussed in more details below and commented with some specific robotic application standpoint.

Data rate performance

With a typical range [100-200 Mbps] for SpaceWire and a demonstrated 2.5 Gbps line rate with current flight technology, 6.25 Gbps line rate planned, and a demonstrated Multilaning x 4 up to 10 Gbps with the SUNRISE project shown at DASIA 2016 [30][31], the SpaceWire/SpaceFibre system completely covers the robotics applications needs for e.g. controlling a robotic arm while carrying visual or radar sensor data.

Traffic policy

There are noticeable improvements in the SpaceFibre from the current SpaceWire standard such as the split of packets into frames of limited size with three associated levels of integrated Quality of Service (QoS) based on Priorities, Bandwidth Credit Precedence and Scheduled traffic. Also, both at Node and Routing Switch ports level, the definition of Virtual Channels allows creating segregated routes within the same wire so that a SpaceWire packet flowing through one Virtual Channel does not block another packet flowing through another Virtual Channel.

Scalability

The 32 Virtual Channels (VCs) per switch or node allowed by the SpaceFibre specification is probably enough for simple robotic applications (e.g. robotic arm). It will be a limitation in larger systems with many effectors, potentially interconnected and implementing a strong scheduling policy for time criticality purpose. Workaround in robotics systems will be a physical split in several networks which may increase the complexity of the architecture and system control software. This limitation also becomes a problem when using Time and Space Partitioning (TSP) execution platform as expected for most critical applications in the future (see below in this paper).

Determinism

Bounded Latency and Jitter properties are inherited from SpW, i.e. variable based on topology, data volume and speed of links, estimated $\leqslant$10 µs in most cases. Network real-time analysis will be required for most use cases which should be eased through efficient tools for representative network modelling and simulation. Analysis of the network topology and configurability will allow performing network calculus to mathematically compute the maximum latency and jitters in nominal cases. But worst case computation of latency/jitter so that to avoid overflow is tricky: indeed, it becomes extremely challenging to verify/guarantee that the system is schedulable when QoS features such as automatic retries are taken into account. Tools based on schedulability theories can help here. There is currently no network calculus model or tool suite as commercially available product. SpaceWire and SpaceFibre modelling is possible to a certain extend with MOST [32] based on the OPNET® network simulation system which could be a basis for further development.

Synchronisation and time management

In the current SpaceFibre specification, time stamping, clock synchronization and global time Distribution keeps the same approach thus the same limitations than for SpaceWire: time codes not clearly defined in current version of the specification but interpreted as covered through the existence of short high priority broadcast frames. Routing of those frames at switch level is neither described nor specified and the message broadcasting is still to be defined. Covering such functions in a standard way for SpaceWire-2 would be useful to robotic systems as well as for many other real-time and autonomous space applications concerned about synchronization of distant units such as sensors, actuators, effectors and computers.

Fault containment

There is no fault containment at VC level in a SpaceFibre node: a failure detection at node level leads to reset the whole faulty link which becomes silent if no auto-start is configured.

There are also no bus guardian features described that could avoid fault propagation if implemented at switch level (e.g. such as with SpW-FDIR). However, some fault propagation may be prevented through the check that VC is properly

configured in the switch or node when routing from the input port to the output port. There is also no standard traffic policy provided based on bandwidth allocation per VC for instance.

BABBLING IDIOT AVOIDANCE

With SpaceFibre, several mechanisms exist allowing the detection and preventing the propagation of errors with babbling idiots:

- erroneous application behavior can be detected through VC buffers allocations and QoS;
- failure of a node, End-System (E/S) or switch, may be detected through sequence number monitoring for instance in case of repeated emission of the same frame or emission of a spurious frame;
- failure of QoS management may be detected at the E/S node level only (no verification at switch level).

NETWORK MANAGEMENT

SpaceFibre specification lacks the definition of standard configuration tables, process and usage domain. Configuration data of the End-Systems can be specified as part of the Management Information Base but configuration of Routing Switches without traffic policy are deemed to be static (generated offline) equivalent to SpaceWire configuration. This does not help for defining dynamic routing, network discovery or autonomous FDIR which could be useful in some robotics systems. The specification of a standard network management also covering network FDIR such as the N-MaSS as described in section V above would be required for SpaceWire-2.

COMPATIBILITY WITH TIME AND SPACE PARTITIONING

Future execution platforms for on-board critical systems will feature Time and Space Partitioning (TSP) allowing functions with different levels of criticality to securely share the same computing and network resources [33]. This will also be beneficial to robotic systems. SpaceWire and SpaceFibre are not exactly designed to be *TSP friendly* – as for instance AFDX is [34]. Virtual Channels and QoS policies introduced at data link layer for SpaceFibre are useful but some design limitations and implementation choices deserve some specific attention as they can jeopardize TSP compliance.

The limitation to a maximum number of 32 Virtual Channels (Clause 5.7.2.b, page 130 in [2]) is probably too restrictive for Time and Space Partitioned systems. Indeed the Virtual Channel concept implies a usage domain where at most (VC number-1) equipment can be configured to send messages to the same equipment without compromising the QoS guarantees between them. Similarly the VCs allocation applied to a TSP system would limit the number of partitions per equipment to the number of available Virtual Channels for a given link and increase the combinatory expansion of VCs allocations. The necessity to reserve point-to-point VCs for preserving partitioning properties with Peer to Peer connections comes from the packet switching inherited from SpaceWire: for a given port, a given VC remains occupied as long as the current packet transmission is not completely transmitted. SpaceWire packet being of potentially infinite length, sharing a VC at switch port level is then not compliant with the partitioning concept.

The number of Virtual Channels necessary to achieve such point to point partitioned connections between nodes (switch ports), can be expressed as

$$F(n) = 2^{\log2(n-1)+1} - 1 \text{ for } n > 1 \qquad (1)$$

This is plotted in Fig. 15 and illustrated with 5 equipment units on Fig. 16 using a SUNRISE router [30][31] offering 4 VCs. We can only allow defining strong QoS policies between 4 non-TSP equipment units, and adding a fifth equipment unit would require 7 VCs.

The frame sequence numbering is not segregated per Virtual Channels but is managed at lane/link level, impacting all messages. Loss of one message is a source of timing interference for all packets/frames.

The routing at switch level is based on packet routing: sharing of a Virtual Channel between two switches' input ports is not recommended as there is no control on packet length and no input port round robin as long as a packet is still in progress for the given Virtual Channel.

The routing switch configuration policy and parameters are not defined yet, assuming that routing is performed through legacy SpaceWire addressing (logical or physical). Routing interference is under the sole control of each application at packet building time with no guardian or traffic policing at switch level. This breaks a little bit the benefits of partitioning.

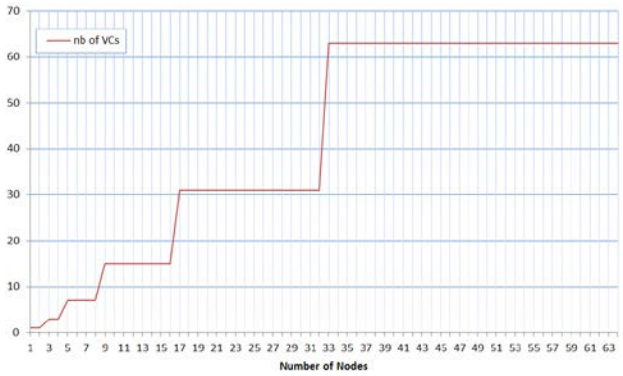


Fig. 15 – Minimum number of Virtual Channels to guarantee the existence of QoS controlled connections between all nodes

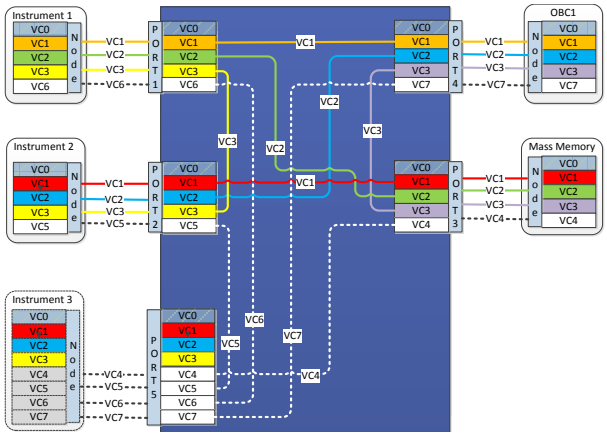


Fig. 16 – SpaceFibre Virtual Channels configuration vs. partitioning

VII. Way Forward and Conclusion

The future space exploration and in-orbit operation projects foresee a more intensive use of robotics systems featuring more autonomous and accurate skills. As this happens more and more with ground applications, also human in-orbit operations and exploration toward the Moon or Mars will benefit from the assistance of smarter robots. One of the main stakes in the near future is the near-Earth active debris removal addressed in the *Clean Space* initiative of ESA [35].

To face this challenge, technology evolution is necessary as classical and reliable datalinks such as the MIL-STD-1553 or the CAN bus become clearly limited with bandwidth capacity below 1Mbps for carrying images or other data intensive sensors, which will require performance in the range of 100 Mbps or higher. Robotics in ground application typically use Ethernet based solutions such as the popular EtherCAT [36], an Ethernet based fieldbus standard which provides high bandwidth with an excellent real-time response for command and control as well as deterministic properties ensuring the level of safety required for working in a human environment. EtherCAT technology has also been successfully used for a robotic arm demonstrator within preparatory studies for the *Deutsche Orbitale Servicing Mission* project (DEOS) [37].

For space applications, SpaceWire and SpaceFibre have the required high bandwidth and low latency capability and could be a good candidate for future robotics with the advantage of being a standard already used for space payload and platforms data handling. However, some aspects of the SpaceWire standard and the SpaceFibre draft standard are not directly adapted to robotics specific needs such as, for instance connectors, cables or error management for FDIR and a few functional aspects that could be corrected. There are also identified drawbacks that may prevent evolution toward larger and more complex systems than just robotic arms, capture mechanisms or rovers that may emerge in the future. A SpaceWire-2 standard covering an evolution of the current SpaceWire and SpaceFibre networks would be beneficial if, in addition to the coverage of future platform and payload needs, it could take into account the specific needs of space robotics including margin for evolutions for the benefit of the future development of space exploration and in-orbit operations.

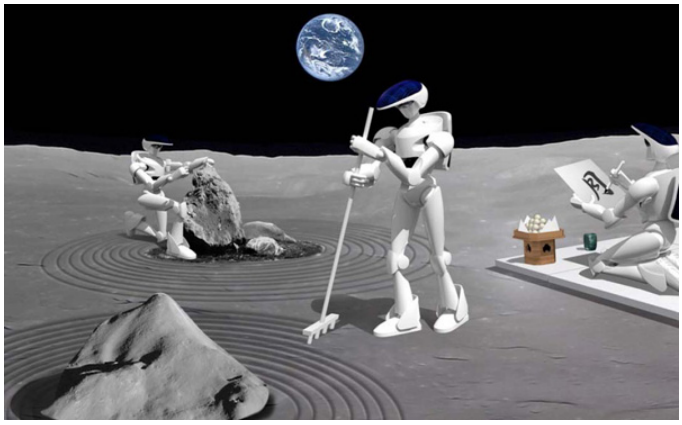


Fig. 17 – Space robotics: One Giant leap for Robots
SOURCE: www.robotzeitgeist.com [37]

References

[1] ECSS Standard ECSS-E-ST-50-12C; SpaceWire, Links, Nodes, Routers and Networks, European Cooperation for Space Data Standardization, July 2008

[2] Steve Parkes, Albert Ferrer, Alberto Gonzalez, Chris McClements (University of Dundee); SpaceFibre Specification, Draft H4: April 2016

[3] *SpaceWire Standardisation planning*, presented by Martin Suess (ESTEC) at the 21st SpaceWire Working Group Meeting, in ESTEC, Noordwijk, The Netherlands on 17th September 2013

[4] *SAVOIR-UNION status*, presented by Christophe. Honvault (ESTEC) at ADCSS in ESTEC, Noordwijk, The Netherlands on 20th October 2015

[5] *Performance evaluation of SpaceWire-R IP Core and updates of the specification*, presented by Dr. Takayuki Yuasa (RIKEN) at the 24th SpaceWire Working Group Meeting in ESTEC, Noordwijk, The Netherlands on 23rd September 2015

[6] *SpaceWire-R implementation and test considerations*, presented by Krzysztop Romanowski (ITTI) at the 24th SpaceWire Working Group Meeting in ESTEC, Noordwijk, The Netherlands on 23rd September 2015

[7] *SpaceWire-D User Requirements* presented by Jérôme Lachaize and Clément Pruvost (Astrium) at the 21st SpaceWire Working Group Meeting in ESTEC, Noordwijk, The Netherlands on 18th September 2013

[8] Steve Parkes, David Gibson - University of Dundee, Space Technology Centre and Albert Ferrer - STAR-Dundee Ltd, United Kingdom; *SpaceWire-D: Deterministic Data Delivery over SpaceWire*, DASIA Conference, Warsaw, Poland, June 2014

[9] *SpaceWire-RT Project Results*, presented by Steve Parkes (University of Dundee) at the 21st SpaceWire Working Group Meeting in Noordwijk, The Netherlands, 17-19 September 2013

[10] *Network Discovery Protocols Final Presentation*, presented by Stuart Fowell (SciSys) at the TEC-ED and TEC-SW Final Presentation Days in Noordwijk, The Netherlands, 10th December 2014,

[11] Giuseppe Montano - Astrium Ltd, United Kingdom, David Jameux – ESA/ESTEC, The Netherlands, Barry Cook, Roger Peel, Ecaterina McCormick, Paul Walker - 4Links Ltd, United Kingdom, Vangelis Kollias, Nikos Pogkas - Teletel, Greece; Network Management and FDIR for SpaceWire Networks (N-MaSS), DASIA Conference, Warsaw, Poland, June 2014

[12] Giuseppe Montano – Airbus Defence and Space Ltd, United Kingdom, David Jameux – ESA/ESTEC, The Netherlands, Barry Cook, Roger Peel, Ecaterina McCormick, Paul Walker - 4Links Ltd, United Kingdom, Vangelis Kollias, Nikos Pogkas, Antonis Tavoularis - Teletel, Greece; Standardisation of the Network Management Service Suite (N-MaSS) for Fault Detection, Isolation and Recovery for SpaceWire, 6th SpaceWire International conference, Athens, Greece, November 2014

[13] ECSS-E-ST-50-12C-Rev1 status, presented by David Jameux (ESTEC) at the 24th SpaceWire Working Group Meeting in Noordwijk, The Netherlands on 22nd September 2015 |

[14] Steve Parkes, Chris McClements - University of Dundee, Space Technology Centre, United Kingdom, Albert Ferre, Alberto Gonzalez Villafranca - STAR-Dundee, United Kingdom; SpaceFibre Implementation, Test and Validation, DASIA Conference, Warsaw, Poland, June 2014

[15] Olivier Notebaert, Jérôme Lachaize, Rémi Clavier, Andre Fueser, Hans-Juergen Herpel, Giuseppe Montano, Luc Planche, Airbus Defence and Space France, UK and Germany; SpaceWire-2: needs and evaluation metrics, 6th SpaceWire International conference, Athens, Greece, November 2014

[16] Alain Benoit - ESA/ESTEC, The Netherlands; Space Avionics open Interface Architecture (SAVOIR), First European Space Technology Harmonisation Conference, ESA/ESTEC, The Netherlands, 18-19 March 2014

[17] Christophe Honvault – ESA/ESTEC, The Netherlands; Avionics – Embedded Systems / On-Board Software, First European Space Technology Harmonisation Conference, ESA/ESTEC, The Netherlands, 18-19 March 2014

[18] ESA/ESTEC, e.deorbit Implementation Plan, Issue 1.0, 18/12/2015

[19] Airbus Defence and Space e.Deorbit Phase B1 results presentations at the Clean Space Industrials days in ESTEC, Noordwijk, The Netherlands on 23rd to 27th May 2016, https://indico.esa.int/indico/event/128/

[20] Clean Space Team (ESA/ESTEC), CleanSat presentation at the Clean Space Industrials days in ESTEC, Noordwijk, The Netherlands on 24th May 2016

[21] Michel Frezet (Airbus Defence and Space), Space Tug Presentation at the Clean Space Industrials days in ESTEC, Noordwijk, The Netherlands on 24th May 2016

[22] *Automation and Robotics within the German Space Program & the DEOS Mission*, presented by Bernd Sommer at the European Conference on On-Orbit Satellite Servicing and Active Debris Removals in Brussels, Belgium, October 30th 2012

[23] Jordi Artigas (DLR), The OOS-SIM: An On-ground Simulation Facility For On-Orbit Servicing Robotic Operations, 2015 IEEE International Conference on Robotics and Automation (ICRA) Washington State Convention Center, Seattle, Washington, May 26-30, 2015

[24] Dr. Gordon Roesler (DARPA) Robotic Servicing of Geosynchronous Satellites (RSGS) Program Overview, Future In-Space Operations Colloquium, June 15, 2016

[25] Axon cable and interconnect: High speed links, www.axon-cable.com/publications/HIGH-SPEED-LINKS.pdf

[26] www.axon-cable.com/en/02_products/03_connectors/04

[27] www.honeybeerobotics.com/portfolio/dust-tolerant-connector/

[28] Marek Rucinski, Adam Coates, Giuseppe Montano, Elie Allouis, Airbus Defence & Space; David Jameux ESA-ESTEC; SpaceWire-based Control System Architecture for the Lightweight Advanced Robotic Arm Demonstrator (LARAD), DASIA Conference, Barcelona, Spain, May 2015

[29] Network Latency Analysis of a SpaceWire-based Control System for Space Robotic Arm; Giuseppe Montano, Marek Rucinski, Elie Allouis, Olivier Notebaert, David Jameux | International SpaceWire Conference | 25-28/10/2016

[30] Steve Parkes, Chris McClements, David McLaren – University of Dundee, Scotland, UK, Albert Ferrer Florit, Alberto Gonzalez Villafranca, STAR-Dundee Ltd, Scotland, UK, SpaceFibre: The Standard and the Multi-Lane Layer, DASIA Conference, Tallinn, Estonia, May 2016

[31] Steve Parkes, Chris McClements, David McLaren, University of Dundee; Albert Ferrer Florit and Alberto Gonzalez Villafranca STAR-Dundee Ltd; SUNRISE: A SpaceFibre Router, DASIA Conference, Tallinn, Estonia, May 2016

[32] Brice Dellandrea, Thales Alenia Space, France; Steve Parkes, University of Dundee, United Kingdom; David Jameux, ESA/ESTEC, The Netherlands; MOST: Modeling of SpaceWire & SpaceFibre traffic, 6th SpaceWire International conference, Athens, Greece, November 2014

[33] *SAVOIR-IMA - Integrated Modular Avionics and Time and Space Partitioning*, SAVOIR report: TEC-SW/09-247/JW available at https://essr.esa.int/

[34] Marie-Hélène Deredempt - Astrium, France, Vangelis Kollias - Teletel, Greece, Zhili Sun - University of Surrey, United Kingdom, Ernest Canamares - GTD, Spain, Philippe Ricco - CES, Switzerland; An AFDX Network for Spacecraft Data Handling, DASIA Conference, Warsaw, Poland, May 2014

[35] www.esa.int/Our_Activities/Space_Engineering_Technology/Clean_Space

[36] EtherCAT, the Ethernet fieldbus, https://www.ethercat.org

[37] P. Rank, Q. Mühlbauer, Kayser-Threde; W. Naumann, SpaceTech; K. Landzettel, Institute of Robotics and Mechatronics DLR; The DEOS automation and robotics payload, ASTRA conference, ESA/ESTEC, Noordwijk, The Netherlands, 12-14 April 2011

[38] Robotics Zeitgeist - Artificial Intelligence and Robotics blog, *Space robotics: One Giant leap for Robots* http://robotzeitgeist.com/tag/mars-rovers

BepiColombo - building a robust Data Management Subsystem utilising SpaceWire networks

Missions and Applications, Long Paper

James Windsor
Wahida Gasti
European Space Research and Technology Centre
Noordwijk, The Netherlands
James.Windsor@esa.int, Wahida.Gasti@esa.int

Ignacio Clerigo
European Space Operations Centre
Darmstadt, Germany
Ignacio.Clerigo@esa.int

***Abstract*— This paper presents the approach used by the BepiColombo data management subsystem and its SpaceWire network to handle system resets following major anomalies and to ensure availability of communication with the payload and platform units. The system autonomy requirements are presented with a discussion on how they drive the robust implementation of the data management subsystem. Two on-board SpaceWire networks are presented accompanied by an overview of the SpaceWire links, the link failure scenarios, and a description of the functions needed to manage the network. The network management functions are mapped to specific nodes in the SpaceWire network which enables the data management subsystem to initialize the network, detect failures, perform recovery actions, ensure network availability for critical nodes; and that the data transmission performance is acceptable with specific protection against excessive data transfer to the on-board computer. The role of each unit in the SpaceWire network and their contribution to the overall data management subsystem is described.**

Keywords—SpaceWire; data management subsystem; routers; networks

INTRODUCTION

BepiColombo [1] is an Interdisciplinary Cornerstone Mission to the planet Mercury, in collaboration between ESA and ISAS/JAXA of Japan. It consists of two scientific orbiters, the Mercury Planetary Orbiter (MPO) and the Mercury Magnetospheric Orbiter (MMO), which are dedicated to the detailed study of the planet and of its magnetosphere. The system baseline envisages the launch of the MPO and MMO composite spacecraft on an Ariane 5 vehicle. The nominal duration of the Interplanetary Cruise Phase (from launch until insertion into the operational Mercury orbit) will be approximately 6 years and will involve a combination of a number of Earth, Venus and Mercury gravity assists, and Solar Electric Propulsion. The nominal MPO operational life will be at least 1 Earth year in Mercury orbit. An extended MPO operational life will be possible of at least 1 Earth year in Mercury orbit beyond the nominal operational life.

The BepiColombo spacecraft system consists of the following modules, as shown in Figure 1:

- Two science spacecraft: The MPO carrying remote sensing, magnetometry and radioscience instrumentation, and the MMO, carrying fields and particles science instrumentation (including the MMO Separation System)
- An MMO Sunshield and Interface Structure (MOSIF)
- A Mercury Transfer Module (MTM)

This paper focuses on the BepiColombo Data Management Subsystem (DMS) with its SpaceWire network [2], hosted in the MPO spacecraft and the implementation of the mission autonomy.

MISSION AUTONOMY REQUIREMENTS

The mission has five distinct phases, ranging from Launch and Early Orbit, near-Earth commissioning, interplanetary cruise, Mercury approach and finally the Mercury Orbit phase. Over these phases the composition of the spacecraft changes as modules (the MTM, MMO and MOSIF) are separated. The autonomy requirements are driven by the demands of each mission phase, ground visibility and the solar environment.

The requirements address the need for continued mission production generation (i.e. science), nominal operations (platform management), and spacecraft safety when in an emergency. The spacecraft shall be able to continue mission product generation without the need for Ground contact for a period of at least 7 days. The spacecraft shall be able to continue nominal operations without the need for Ground contact for a period of at least 14 days. The spacecraft shall be able to operate in its Safe Mode for the longest expected ground outage (14 days in cruise and 7 days in mission orbit); in addition there are solar conjunction periods of up to 35 days in cruise (with minimal science operations) and up to 10 days in mission orbit (with full science operations), where no ground support is possible and that the spacecraft shall operate in its nominal mode.

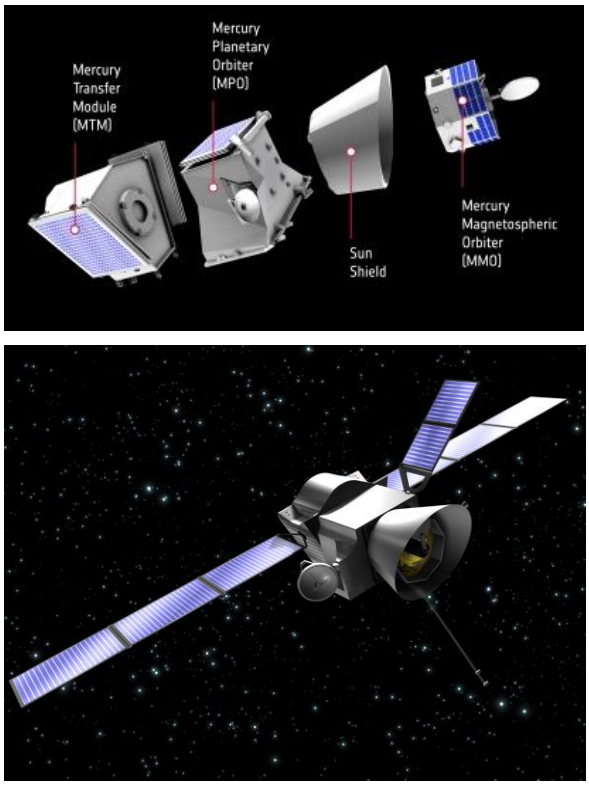


Fig. 1. BepiColombo spacecraft and module composition

In addition to the autonomy duration requirement there are two further constraints. Firstly the system must have attitude control at all times in order to prevent sensitive elements of the spacecraft from being exposed to the sun (e.g. radiators, payload sensors, electric propulsion, and maintenance of solar arrays temperatures within qualification limits). The specification calls for the recovery of any system-level anomaly to be initiated with an on-board computer (OBC) reboot thereby ensuring a clean and consistent system start. A system reboot can take several minutes which would cause unacceptable attitude degradation so the AOCS and DMS together must ensure that attitude control is maintained over any reboot. Secondly, in order to ensure spacecraft safety in any situation, thruster valve commands must be dispatched immediately which means that the data link must guarantee that the thruster commands are received instantaneously and in a coherent manner.

The design of the DMS must be sufficiently adaptive and robust to cope with these requirements, must be resilient to any single failure and must be able to support the autonomy duration requirements. This led to the selection of a SpaceWire network approach as the main communication technology in the DMS design.

DATA MANAGEMENT SUBSYSTEM ARCHITECTURE

At first glance the BepiColombo DMS subsystem is typically of many spacecraft. It is responsible for the storage of generated data, the retrieval and downlink of on-board data, commanding and control over the spacecraft units and subsystems, data acquisition from numerous on-board platform sensors (e.g. thermistors, relays, sun sensors, Star Trackers, IMUs), and the implementation of autonomous functions (e.g. mission timeline, control procedures, parameter monitoring).

The BepiColombo Prime Contractor is Airbus Defence and Space GmbH based at Friedrichshafen, Germany. Airbus DS is responsible for the DMS subsystem, and they designed a subsystem that is functionally split into an architecture for normal operations, and a separate architecture which is only used to recover from a system failure. The solution implemented by Airbus DS is presented in Figure 2 and uses two SpaceWire networks and associated nodes:

- The Payload network connects the Solid State Mass Memory (SSMM) to the On-Board Computer (OBC) and the nine Payload instruments.
- The Failure Control Equipment (FCE) network connects the FCE to the Remote Interface Units (RIU) and the OBC.

SpaceWire was selected as the common payload data link for the command, control and transmission of science data because of severe power and mass constraints on the spacecraft. There are nine payloads hosted on the network with a total mass budget of 60 kg, a total power budget of 140W and data rates in the order of 10 Mbit/s. Previous ESA Science missions had lower numbers of payloads, less demanding data rates with higher total mass and power budgets. BepiColombo needed a solution where a common set of interfaces could be used across all payloads to ease integration, that was compliant to the mass and power budgets, and provided the required data rates. SpaceWire was the only candidate option with only the AT7910E SpW router and SMCS332SpW driver devices available at the time (the avionics design kicked-off in 2007).

As can been seen from Figure 2, the OBC acts as the hub and bridge for the SpaceWire networks. The responsibility for command and control of the networks is distributed across the OBC, SSMM and to a lesser extent the FCE. It is this distribution of responsibility that is the foundation of the network's robustness and adaptiveness.

The nominal AOCS and TT&C subsystems use a redundant MIL-STD-1553B data bus for command and control over the remainder of the S/C units (e.g. StarTrackers, Inertial Measurement Units, Deep Space Transponder). Communication with the MMO is also via MIL-STD-1553B.

A. Network Redundancy Management

The DMS uses a nominal and redundant unit management approach. The units are either physically separated (e.g. separate instruments) or the single unit internally houses a nominal and redundant side, examples of the latter are the OBC, FCE, SSMM and the two RIUs. Each side of these units is then connected to the SpaceWire network as shown in Figure 2. Unlike other data links such as the MIL-STD-1553B where there is a nominal and a redundant bus, the SpaceWire network does not have a physically separate redundant network. The nominal and redundant interfaces on the network are cross-strapped which allows for a failed router to be bypassed or for access to a redundant unit via an alternative routing path. In

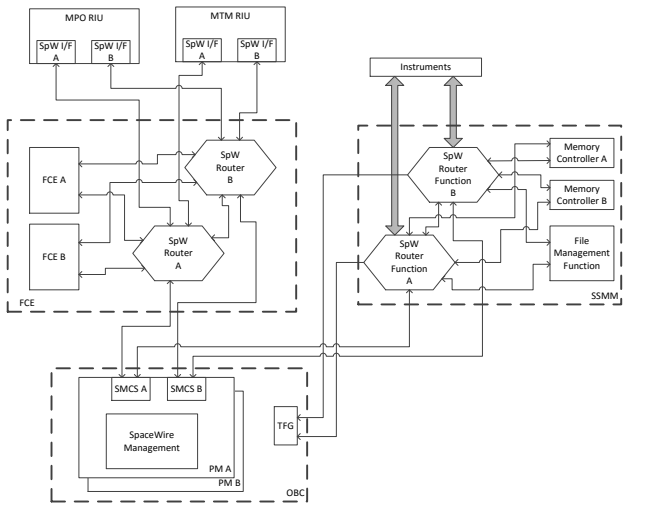


Fig. 2. Data Management Subsystem Architecture with SpaceWire

addition using logical addressing of destinations for SpW messages, rather than physical addressing, separates the addressing from the physical use of the unit redundancy. This means that when commanding or sending telemetry, the transmitter does not need to be aware whether the nominal or redundant destination is in use. The network via the routers will automatically transfer the SpW packets to the correct end destination. The BepiColombo DMS networks build upon the following two foundations: Group Adaptive Routing and project specific extensions to SpaceWire.

1) Group Adaptive Routing

Re-routing a packet out of one of several possible output ports in a SpW router switch is known as group adaptive routing (GAR). Links that connect to the same destination (node or routing switch) are called a group. Any link in a group can be used to transfer data to their destination. GAR is a means of routing packets to a requested destination over different paths through a network and provides a means of managing allocation of link bandwidth ensuring optimised use of the network resources.

Group adaptive routing can be implemented relying on the configuration registers to hold information about equivalent output ports. When a packet is received it can be routed to any of the equivalent output ports that are currently free, or to whichever port become available first. Assignment of a packet to an output port involves also the consideration of any arbitration scheme that is implemented within the routing switch.

Adaptive grouping can remove some tasks from the system design (e.g. understanding which side of a unit is active and actively selecting the target logical address) and it makes the system more robust and transparent to failures; but at the cost of visibility – the network autonomously decides which adaptive branch to take and does not report or record this event which could mask a failure in the network. Another drawback is that GAR potentially lowers the system determinability because different routes are possible which could lead to packet collisions and network congestion.

B. Extensions to SpaceWire Standard

SpaceWire [2] is a full-duplex, point-to-point, serial high speed data link (between 2 Mb/s and 400 Mb/s) for the transmission of payload data and control information on-board a spacecraft. The SpW standard defines a number of levels:

Physical level: defines connectors, cables, cable assemblies and printed circuit board tracks.

Signal level: defines signal encoding, voltage levels, noise margins, and data signalling rates.

Character level: defines the data and control characters used to manage the flow of data across a link.

Exchange level: defines the protocol for link initialisation, flow control to avoid overflow of the receiving buffers, link error detection and link error recovery.

Packet level: defines the packetisation used for transmitted data over a SpaceWire link. The data to be carried across the link is called the SpW cargo. The standard [2] does not define a protocol to be used for this cargo. In the case of BepiColombo there were three cargo protocols adopted as shown in as shown in Figure 3.

- A Packet Utilisation Standard [3] (PUS) compliant single Telemetry (TM) or Telecommand (TC) packet structure was used with the PUS packet carried as cargo in the SpW packet.
- A collection of PUS Telemetry packets bundled together into the cargo field for the OBC-SSMM transfer of data. This maximises the utilisation of the SpW link.
- Remote memory access protocol (RMAP) [5] for the RIU communications to allow direct read (i.e. data acquisition) and write (commanding) access to the RIU memory array.

Network level: defines the structure of a SpaceWire network and the way in which packets are transferred from a source node to a destination node across a network, and how to handle link and network level errors.

Two new layers have to be added to those specified in [2]. They are the Application Level and the System Level. They both reside above the network level. The Application Level is hosted at each network node and contains the services for supporting the different types of packet level traffic used on each network:

- PUS packets between FCE and OBC.
- RMAP exchange is used to transmit telecommand and telemetry between the RIUs, OBC, and FCE.
- RMAP to control SpW routers supported by the OBC and SSMM.
- OBC supporting PUS to the payloads and grouped PUS packets to the SSMM.

The System Level covers the management of the SpW networks and the reminder of the spacecraft, e.g. the other networks (e.g. 1553), operators, and subsystems.

C. Impact of SpaceWire on System Determinism

A SpaceWire network by nature is asynchronous. Wormhole routing is used to secure a path via links and routers to ensure that a packet maintains its integrity as it is transported through the network. As per the standard, when a packet arrives at a router it is sent to the allocated output port if that output is free. If it is in use, i.e. congested, then the incoming packet is stalled waiting for the output to be free. Routers do not have extensive memory buffers for storing stalled packets which means the packet is stalled from the point in the network where the collision has occurred back to the transmitter device. This wormhole along the network stalls any other packet that needs to access the congested ports. A similar problem occurs if the destination node is not ready to receive the packet. The packet is stalled on the final link to the destination device and blocks the path back to its source.

This causes a major challenge for providing a deterministic communication over a SpW network. Data has to be able to flow through the network and when the data arrives the application level must process it sufficiently quick enough to allow new packets to arrive. Currently the only solution to this challenge is to design the network communication to have sufficient bandwidth margin to avoid congestion and to ensure that the application software can rapidly process incoming SpW packets. The application software must take into account the asynchronosity to avoid the data bus from pre-empting critical system software tasks and upsetting the system's schedulability.

D. SpaceWire Fault Tolerant Links

SpaceWire is based on a fairly robust physical layer based on a driver/receiver pair and a shielded cable with very good EMC properties (BER = 10^{-12}). If a transient error does occur then the SpaceWire Codec immediately disconnects the link electrically and goes through the re-initialisation process. In 20 µs the link is up and running again. The packet that was in the process of being transferred is truncated and terminated by a special Error End of Packet (EEP) character to indicate that it was terminated prematurely.

The System Level is then responsible for managing the EEP to resend or request the missing packet. Protocols for providing this type of service are thus at system level.

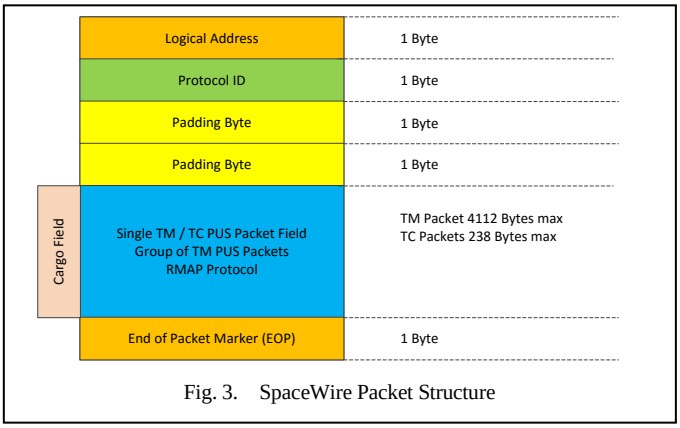


Fig. 3. SpaceWire Packet Structure

E. SpaceWire failures

It is possible to derive a number of failure definitions that are generic to a SpaceWire network implementation based on the failure modes identified in the SpW standard.

1) Link Errors

The SpW standard defines clear cases, such as due to parity errors in the transmitted data, when a link should be restarted either by the transmitter or the receiver end. The standard defines at exchange level the failure detection and recovery approach to be followed at device level, i.e. by either the AT7910E SpW routers and SMCS332SpW drivers, and when the error should be reported to a system level, as shown in Figure 4.

2) Router failure

If the router fails to work then the symptoms are either continued link disconnects on the nodes connected to the device, the disappearance of SpW packets from the system because the router is losing them, or data corruption with invalid packets or timecodes.

The network level cannot detect a router failure. It is only at system level, or spacecraft operator level, that all of the symptoms can be detected, consolidated and a correct diagnosis made. A router failure can be automatically bypassed if Group Adaptive Routing is enabled on the SpW routers. This allows all routers connected to the failed router to detect an unavailable link and use an alternative routing path to transfer packets but without reporting the anomaly.

3) Node failure

If the node connected to the network has failed then the failure symptoms are continued link disconnects with the connected driver or router, or the unit does not send any packets on the network. SpaceWire does not have an acknowledge protocol for non-RMAP traffic so if a packet has been lost on the network, e.g. due to a router failure or a link disconnect, then there is no indication at network level that the packet is missing. The only indication can be at system level when either the commanded behaviour is not observed or telemetry is not received from the node.

A complication arises that a router failure provides similar symptoms as a node failure (missing packets) so it is necessary when diagnosing a potential failure that the complete SpW path is checked.

4) Missing data

Missing data can only be detected by missing telemetry reports that are sent upon request by a user, or a failure for a telecommand to change the on-board state, or by the ground operators due to jumps in the telemetry source sequence counts. It will be unclear what is the cause of the anomaly. It could be any of the failure cases (link errors, router or node failures) and the cause of the anomaly can only be determined at system level through a process of elimination.

If the RMAP protocol [5] is used then each communication on the network becomes a message exchange. A write

command results in a the destination node sending a copy of the write result back to the transmitter. A read command results in the read data being returned. This protocol allows missing data to be detected without providing the root cause.

SPACEWIRE NETWORK FUNCTIONS

Based on the previously identified network failure cases, it is possible to further derive a set of functions that are need by the DMS in order to effectively manage the network are presented in the following subsections. For BepiColombo these functions are distributed across the DMS. A mapping of where the network management functions are implemented is provided in Table 1.

F. Network Initialisation

This function is responsible for ensuring that the network is powered in the correct sequence, the node SpW specific self-tests are performed, and the SpW links are running (i.e. exchange of NULL tokens).

G. Network (re-)configuration

This system level function is responsible for selecting the nominal or the redundant interfaces of the network, which are defined in terms of nodes, routers and links.

A network reconfiguration shall be a failure recovery following an anomaly in either the network's nominal (or redundant) interface or a failure at payload level which necessities a reconfiguration in order to access the redundant payload. It is possible to have a full reconfiguration which is a complete switch the complete set of interfaces, or a partial reconfiguration where cross-strapping is used to replace only those failed elements of the network with their redundant elements.

H. Router (re-)configuration

This network level function programmes the router using dedicated SpW commands. Routers can be configured at any time as long as the router is powered.

A router reconfiguration shall be needed if a link has failed and its port allocation needs to be removed or its behaviour needs to be modified.

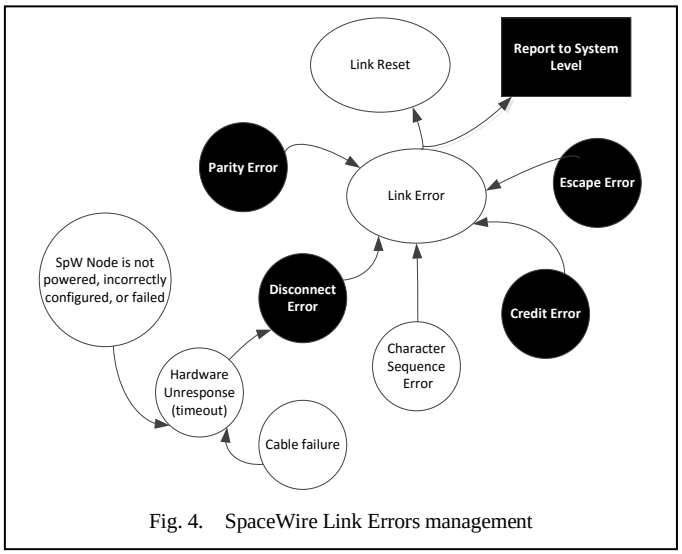


Fig. 4. SpaceWire Link Errors management

I. Router health monitoring

There is no dedicated health reporting provided by the AT7910E router so the system level must contain a function which is able to verify the health of the routers and to ensure that they are correctly configured.

J. Link health monitoring and SpaceWire packet error detection

The AT7910E router detects and recovers from link errors on each of its ports and sets an error register. This is implemented at network level, however there is no convenient means to report this to the system level. A dedicated function has to be added to the network level to access the router error registers in order to flag link errors to the application layer.

The SMCS332SpW driver will detect and recover from a link error while raising an interrupt to the application layer.

K. System Traffic volume and data flow protection

In the SpaceWire standard flow control is implemented at exchange level and ensures data is only transmitted when there is available space at the receiver buffer. This is a low level implementation and has no knowledge if the receiving application, or node, is ready for the data. Sending excessive volumes of data to a destination node in a so called 'babbling idiot' [6], essentially an unintentional Denial of Service attack, can lead to network performance degradation and in the worst case, a severe impact on the performance of the receiving node. In the case of BepiColombo the receiving node is the central OBC meaning any degradation in the unit's performance due to excessive incoming SpW packets will endanger the mission safety. A system function is needed to implement a network level flow control mechanism because it is not supported in the SpaceWire standard.

PAYLOAD NETWORK

The Payload Network has its external SpW links configured to be 10 Mbit/s between the SSMM, OBC and payloads, with one exception for a payload-SSMM link which is configured to be 100 Mbit/s. The network has several purposes:

- It must transfer commands and TM from the OBC to the SSMM for processing and storage.
- The OBC must receive housekeeping telemetry from the SSMM and payloads for monitoring, and it must be able to command the nine payload nodes. The SSMM must receive both Science data (stored directly in the packet stores hosted inside the SSMM) and forward the non-Science housekeeping telemetry to the OBC.
- The OBC distributes the time throughout the network.
- To enable the downlink of data from the packet stores within the SSMM via the Telemetry module built into the OBC.

The SSMM is defined in full detail in [4]. The SSMM is supplied by Thales Alenia Space – Italy, Milan and has itself an embedded SpaceWire network running at 20 Mbit/s based

on ten AT7910E SpW routers ASICs connecting three SpW nodes (the memory, Supervisor A and Supervisor B) to the external nodes (nine instruments, OBC, Telemetry modules, and test equipment). The unit provides the command, control and time code distribution link between the OBC and the payloads.

As shown in Table 1, the majority of the network functions are hosted in the SSMM. This was done to lower the work load on the system and ground levels and to centralised all of the router configurations. The nominal and redundant Supervisor modules each house an ERC32 TSC695F microprocessor that run the unit's application software (ASW) which implements all of the allocated network functions. The unit maintains a copy of its internal state and configuration in non-volatile memory so that following any recoveries from minor anomalies the unit can restore itself to its previous setup. Any major anomalies will lead the unit to enter its Service mode. In this case the unit management hosted in the OBC will then reconfigure the SSMM supervisor to its redundant side if not already in use to permit continued operations.

A. Network Initialisation

The SSMM initialises the network such that the routers that interface with the payloads, and the payloads themselves, have the 'link-start' configuration meaning that immediately following power-on they attempt to start the link. This ensures that if the payloads are ready to transmit then the network is configured to receive the data. The side of the network not used (e.g. the redundant side) is disabled as part of the initialisation logic. This prevents any SpW links running on the non-used sides which could happen when both the nominal and redundant sides of the receiver (e.g. SSMM) and both the nominal and redundant sides of the transmitter (e.g. Payloads) are cross-strapped and incorrectly configured.

B. Network Configuration and Health Monitoring

The OBC is supplied by Thales Alenia Space – Italy, Milan and is based on the Leonardo architecture which has heritage from other ESA missions such as CRYOSAT and SWARM. The OBC uses a ERC-32 based Processor Module architecture and supports data exchange via SpW using the SMCS332SpW driver to interface to the network. The OBC is the source of all commands dispatched into the network and the destination of all non-science telemetry generated by the payloads and SSMM. The system autonomy and monitoring is implemented within the OBC. The SSMM management function is implemented in the OBC and is responsible for ensuring the availability of the SSMM and the Payload network. The SSMM-OBC links are cross-trapped and the OBC selects the link and takes care of reconfiguring it in case of failure.

It is necessary for the SSMM to be fully aware of all possible router configurations in order to ensure that the network can be reconfigured to ensure the command, control and time link between the OBC and the payloads. The unit can detect SpW link errors and notify the OBC. The OBC will then determine if further correct action is needed (the SSMM will automatically restart the SpW link as per the standard) otherwise the OBC will record the event for further consideration by system level failure management functions and spacecraft ground operators.

If at either system or spacecraft operations level it has been decided to reconfigure the network, then commands are sent to the SSMM notifying it of the new external interfaces to use and also any payload commands are sent to switch over to the redundant (or back to nominal) sides. The OBC does not specify which router to select or what the programming is to be. It simply commands select Payload Interface Nominal or Redundant. The SSMM is aware of the state/health of its network and is able to take the necessary steps to reconfigure the routers to ensure that the OBC has the requested command, control and time link. It is this level of decentralised control at SSMM level that reduces the complexity of the DMS subsystem, thereby improving its robustness.

The only approach for the SSMM to verify the health status of its routers is to send an RMAP interrogation command requesting the status of certain registers. There is not a dedicated register reporting the overall health of the router, meaning it is not possible to poll a single entry instead several entries must be checked and then compared against previous

TABLE I. SPACEWIRE NETWORK MANAGEMENT FUNCTIONS

SpaceWire Function	**Payload network (OBC, SSMM, Payloads)**	**FCE network (OBC, FCE, RIUs)**
Network initialisation	SSMM, OBC, Payloads	OBC, FCE, RIU.
Network (re-)configuration	OBC sends commands to SSMM which then performs the configuration activities on the payload interfaces. The SSMM is capable of automatically configuring its output interfaces to the Telemetry Frame Generators for downlink of data.	OBC. FCE – only its inteface to router.
Router (re-)configuration	SSMM. All direct configuration activities are localised to the SSMM. This ensures that only the SSMM needs to be aware of the different configuration options thereby lowering the effort needed at system level.	OBC.
Router health monitoring	Router health check is only during SSMM intialisation.	OBC.
Link health monitoring and SpW packet error detection	SSMM – monitors its own routers and reports errors to the OBC. OBC – monitors its own interfaces. Reacts to errors reported to it and its own errors. Payloads – monitors its own interfaces and reports errors to the OBC.	OBC. FCE – only its link to the router.
System Traffic volume and data flow protection	SSMM – caching function to smooth out bursts of traffic.	None.
Group Adaptive Routing	Used for load balancing traffic transferred into the memory management controller.	Used for managing link errors.

values in order to find an anomaly (frozen or out-of-range value). This approach has not been implemented on the Payload Network because it would add additional traffic – the RMAP protocol is a command-and-response message meaning each register check would add two messages for each of the ten routers to the overall communication budget.

C. Volume and Data Flow Protection

The System Traffic volume and data flow protection for this network is implemented in the SSMM as a caching function. This prevents a babbling instrument from swamping the OBC with SpW exchanges. The SSMM has a pre-defined number of packets that it can transmit per second and it limits the exchange of non-Science telemetry packets to that limit. The SSMM can detect an excessive accumulated packet rate and will report this to the OBC. The SSMM is not able to read the contents of the packets and so is unable to determine the culprit(s) for the excessive packet generation rate. Implementing such an ability would have required dedicated firmware within the unit because the AT7910E router does not support such a monitoring. Examining the content of each packet would have been detrimental to the overall goal of fast SpW operations. It then becomes the responsibility of the spacecraft operators to analysis the available data and determine the culprit.

D. Network Design Driver

The mission critical design driver of the network is to ensure reception of the non-Science telemetry from the payloads in order to ensure their safety. The on/off power commanding of the payloads is via dedicated control lines in the Power subsystem and is independent to the SpW link. During a spacecraft emergency the instruments need to be switched off in a controlled manner. Immediately removing power from several of the instruments could lead to damage because a power-down cycle is needed before cutting power. In addition, an unexpected system failure recovery will involve thruster actuation leading to a risk of internal misalignment of sensitive surfaces and sensors within several of the instruments, therefore these sensitive instruments must enter a safe mode to protect themselves. Due to the asynchronous nature of SpaceWire there is no guarantee that a ‘enter safe mode’ command sent by a system level failure manager to each payload notifying them that a system restart is about to be triggered will reach the destinations, so a different approach was adopted that builds on the SpW timecode mechanism.

The network uses the one second SpW timecode pulse as the heartbeat of the OBC-SSMM-Payload network. The timecode is sent by the OBC and is absent during a system reconfiguration. Therefore if an instrument does not receive a timecode, then it starts to prepare to enter its safe mode. A drawback to this approach means a failure at router or link level could lead to an unintentional safeing of the instruments. To avoid this, a consecutive number of timecodes must have been missed before the instruments take action.

FAILURE CONTROL EQUIPMENT NETWORK

The FCE network is purely a command and control network with no payload involvement. It connects the two flight computers, the On-Board Computer and the Failure Control Equipment, with the Remote Interface Units. The purpose for the FCE network is to ensure that the spacecraft attitude knowledge and pointing is preserved at all times, including across a system reboot. The FCE network has only four nodes: FCE, OBC, MPO RIU and MTM RIU.

Unlike the Payload network there is no dedicated System Traffic volume and data flow protection implementation on the FCE network. Instead the SpW standard flow control token mechanism is used and the amount of data exchanged between nodes is limited at design level.

A. Network Initialisation

As shown in Table 1 the OBC is the node responsible for the initialisation and configuration of the FCE network and routers. The network initialisation involves the OBC configuring the routers to support group adaptive routing in order to allow RIU communication to automatically be routed correctly to the destination without the system needing to know the health of the network in terms of failed port/links.

B. Network Configuration and Health Monitoring

The OBC monitors the status of its links with the routers. If consecutive transmission or reception errors are detected then the OBC will reconfigure its interface to the routers to avoid using the failed SpW link. There are only two hot redundant routers used on the FCE network and so the assumption has been made that a router failure will be detected as a consecutive series of transmission or reception errors, and so can only be considered as a link error. The only recovery possible is to avoid using the failed link to the router. The adaptive routing and the router addressing table ensures that the commands reach the correct destination without needing any router re-configurations. The health monitoring of the router is performed by the OBC polling router registers via RMAP read commands and the values reported in telemetry and available in the datapool. Note that this approach is not used on the payload network due to the number of routers involved (Payload has 10 routers versus 2 with the FCE). Failures could then be diagnosed and acted upon at system level or by ground operators.

The FCE is supplied by Thales Alenia Space – Italy, Milan and is a derivative of the OBC with an additional module to house the routers and access to dedicated sensors. The FCE does not configure the router or the network but it does monitor the status of its links with the routers. If consecutive transmission or reception errors are detected then the FCE will reconfigure its interface to the routers to avoid using the failed SpW link. The adaptive routing and the router addressing table ensures that the commands reach the correct destination without needing any router re-configurations. This routing approach provides the adaptiveness needed at network level to handle failures and ensure the robustness of the subsystem.

The RIUs are supplied by Thales Alenia Space – UK Ltd, Bristol (formerly the Space division of Systems Engineering and Assessment Ltd, UK). The RIUs decouple the discrete

interfaces from the OBC and acts as the central interface units of the satellite. Each RIU contains a nominal and redundant SpW Controller module. Each of these modules (nominal and redundant) has a SpW interface. These signals are cross-strapped to allow both the nominal and redundant SpW controller modules access to the main and redundant SpaceWire links. The cross-strapped signals are passed via a dedicated cross-strap connector which allows the RIU SpW controller to be commanded from either FCE router A or B.

C. *Network Design Driver*

Any system level failure will lead to a system reset requiring an OBC reboot. The duration of the system restart and initialisation is such that any on-going attitude perturbations will cause sensitive surfaces to be exposed to the sun and damaged. To avoid this the FCE takes over control of the RIUs and commands the attitude pointing and platform management. When the OBC is ready to resume spacecraft control, the FCE returns commanding authority and transfers the attitude data to the OBC. Consequently, the key objectives for the network are outlined below:

- Availability of the FCE – OBC link to transfer attitude data back to the OBC following a system reset.
- Availability of the links OBC – RIU and FCE – RIU in order to control thruster actuations.

It is the latter objective that drove the decision to deploy SpaceWire on the FCE network. Using a router allows the master control of the RIUs to be instantly switched from the OBC to the FCE during a system reset. The FCE network is implemented with two AT7910E routers operating in hot redundancy and hosted with their own dedicated power supply as part of the FCE unit. There is no "switch on" or "switch off" service for the router equipment because they are permanently powered. The network is configured to run at 10 Mbit/s except the MTM-RIU link which is at 4 Mbit/s, and all links are initially on autostart mode. As shown in Figure 2 each router unit is able to communicate with all nodes (both OBC PMs, both FCE PMs, both sides of each RIU equipment), but, in the case of the RIUs, the actual cross-strapping is ensured by the bridge link between both routers (router A can only access RIU side A and router B the RIU side B) such that packets have to be forwarded between the routers in order to access the other RIU side. This forwarding is ensured by the adaptive routing mode programmed in both router units. If the direct link to the destination is not active, then the incoming packet is forwarded to the second link as part of the adaptive routing. The second router would then have the active link to the destination node.

LESSONS LEARNT

BepiColombo is the first ESA mission to deploy multiple SpaceWire networks but it has not been an easy adoption of the technology. Fundamental problems have been addressed related to a limited hardware selection, missing key features in the standard [2] (inability of routers to report link errors, no provision of a higher level flow control management where packet rate can be specified per port) which forced workarounds to be implemented at application and system levels. Supporting documentation for the SpaceWire devices and applications at first was poor but evolved over the course of the project, and there was a corresponding lack of previous experience utilising SpaceWire with the spacecraft avionics manufacturers and on-board software developers. Instead a MIL-STD-1553B viewpoint was often applied to SpaceWire which led to inconsistencies between hardware, software and system levels. These issues were discovered and resolved only very late in the System AIT programme. This has led to a firm realisation that it is impossible to build a classical deterministic system in SpaceWire equivalent to a MIL-STD-1553B implementations. It also highlighted a need for training to improve awareness at system level of the advantages and disadvantages of deploying a SpaceWire network on a spacecraft

CONCLUSION

The BepiColombo data management subsystem needed a data link which met stringent mass, power and data rate requirements for the links to the payloads, whilst ensuring an instantaneous switch of RIU control from the OBC to the FCE. A robust data management subsystem has been implemented that is able to autonomously adapt to failures and react to changes in the spacecraft state during system resets incorporating the functions supported by the SpW standard, and implemented in the AT7910E SpW and SMCS332SpW devices. A new set of system level management functions had to be implemented within the OBC and SSMM software in order to coordinate the initialisation of the SpW networks, monitor the network health and take all necessary recovery actions.

BepiColombo is the first ESA mission deploying on-board SpaceWire networks and it has been challenging identifying and implementing the management functions. The industrial team, Airbus DS and Thales Alenia Space, have overcome these challenges and have developed a data management subsystem that meets the needs of this Cornerstone Mission to Mercury.

REFERENCES

[1] Benkhoff, J., van Casteren, J., Hayakawa, H., Fujimoto, M., Laakso, H., Novara M., et al., "BepiColombo - Comprehensive Exploration of Mercury: Mission Overview and Science Goals," Planetary and Space Science, Vol. 58, Issues 1-2, Elsevier, 2010, pp. 2-20.

[2] SpaceWire - Links, nodes, routers and networks ECSS-E-ST-50-12C. 31 July 2008. http://www.ecss.nl/

[3] Telemetry and telecommand packet utilization. ECSS-E-70-41A, 30 January 2003. http://www.ecss.nl/

[4] M. de Meo, G. Saldi, G. Rosani, W. Gasti, J. Noyes, J. Windsor, J. Poeckentrup, R. Eilenberger, "BepiColombo solid state mass memory employing SpaceWire," International SpaceWire Conference 2013.

[5] SpaceWire - Remote memory access protocol ECSS-E-ST-50-52C. 5February2010. http://www.ecss.nl/

[6] I. Broster and A. Burns. The babbling idiot in event triggered real-time systems. In G. Fohler, editor, Proceedings of the Work-In-Progress Session, 22nd IEEE Real-Time Systems Symposium, YCS 337, pages 25–28. IEEE, Department of Computer Science, University of York, 2001.

SpaceWire Fabric Used to Control Family of Standardized High Performance SpaceVPX Modules

Missions and Applications, Long Paper

Joseph R. Marshall, Lisa Assadzadeh, Richard Berger

BAE Systems
Manassas, VA 20112
joe.marshall@baesystems.com

***Abstract*— This paper will describe the various modules and daughter cards under development. Interconnects between these modules will be highlighted with special emphasis on SpaceWire ports, routers and backplane routings. Performance of the modules and their interconnects will be summarized. SpaceVPX Backplanes, cabling and test equipment for bringing up and testing these modules will be described highlighting leveraging of COTS elements.[1]**

***Index Terms*—SpaceWire, Networking, RapidIO, Spacecraft Electronics, SpaceVPX, Ethernet, RAD750 SBC, RAD5545 SBC, RADSPEED DSP, Virtex V5QV, DDR2, DDR3, XR-DIMM, PCI**

I. INTRODUCTION

Future spaceborne systems will require additional onboard processing and much greater interface connectivity. Many efforts worldwide are starting to address these needs. SpaceVPX, a recently released ANSI/VITA standard [1] [2] [3], was created to provide the structure and definition for interoperable modules that will be created to meet these needs. It provides a multi-layer set of fabrics using SERDES, LVDS and LVCMOS devices to provide interconnects in a scalable and fault tolerant way. SpaceWire is setup as both a control plane for command and data handling throughout the box as well as a medium speed data plane.

Building on previous SpaceWire network elements, BAE Systems is creating a set of silicon application specific standard products (ASSP) [4] [5] [6] to provide power efficient general purpose building blocks for the creation of scalable SpaceVPX modules across these three fabrics. These building blocks are key to a new family of SpaceVPX processing and network modules being developed for a wide variety of space applications. These include a RAD750® Single Board Computer [7], a RAD5545™ Single Board Computer, a Virtex-5 based Reconfigurable Computer Module, a dual RADSPEED™ Digital Signal Processor Module and multiple types of standalone function modules focused on memory or unique I/O. All of these modules contain at least dual redundant SpaceWire ports to be utilized by the System Controller for command and data handling or between modules for medium speed data transfers. BAE Systems has also defined daughter cards that may plug into some of these modules which also include SpaceWire links for extending the control, data handling and data transmission. Additionally BAE Systems is exploring ways to interact between Ethernet and SpaceWire for testing and other ground based activities with these modules.

[1] Approved for Public Release – ES-ISR-082216-0107

II. SPACEVPX SYSTEM

SpaceVPX was approved as an ANSI/VITA standard in 2015 and is beginning to see usage across spacecraft systems. It uses three main interface fabrics, RapidIO [8] (up to 20 Gbps per 4 lane port), SpaceWire [9] (up to 400 Mbps per port) and I2C (up to 400 Kbps per port) to provide a scalable interoperable form factor spanning almost five orders of magnitude of performance.

Figure 1 shows a SpaceVPX system made up entirely of SpaceWire links for command and data handling as well as medium speed data movement. The controller module (lower left) directs this through a star topology providing a SpaceWire link to each module. SpaceVPX is also important for its fault tolerance – full single point failure protection is carried throughout the standard. Thus if a system like Figure 1 has redundant modules, the controller would connect to each of those as well. The redundant controller would have an equivalent set of SpaceWire links to each primary and redundant module.

Six instruments are postulated in this system – connections to those instruments may be direct from the controller module, through a remote interface with additional module ports and using heritage or other unique I/O. An internal mass memory is also shown and due to its need to collect or transmit data in larger quantities, it is shown with twice as many connections. This of course could be even higher. Figure 2 shows a slightly expanded system, this time using all three fabrics. As such, the data plane provides significantly more data flow between modules, yet each module still retains it SpaceWire interfaces. Note the ability to connect to instruments using RapidIO, XAUI or other SERDES interfaces and SpaceWire to best

match the performance of each instrument. A data plane switch is also added to avoid a large RapidIO connection mesh.

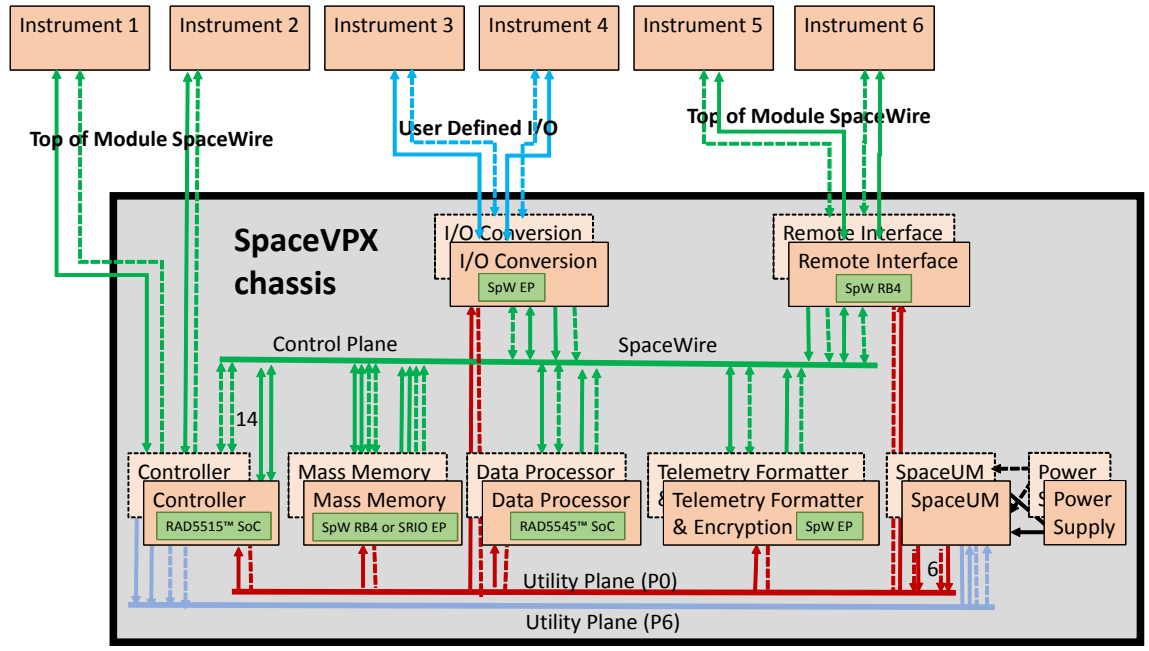


Figure 1: SpaceVPX system using SpaceWire for control and data. Solid horizontal lines represent point to point backplane connections.

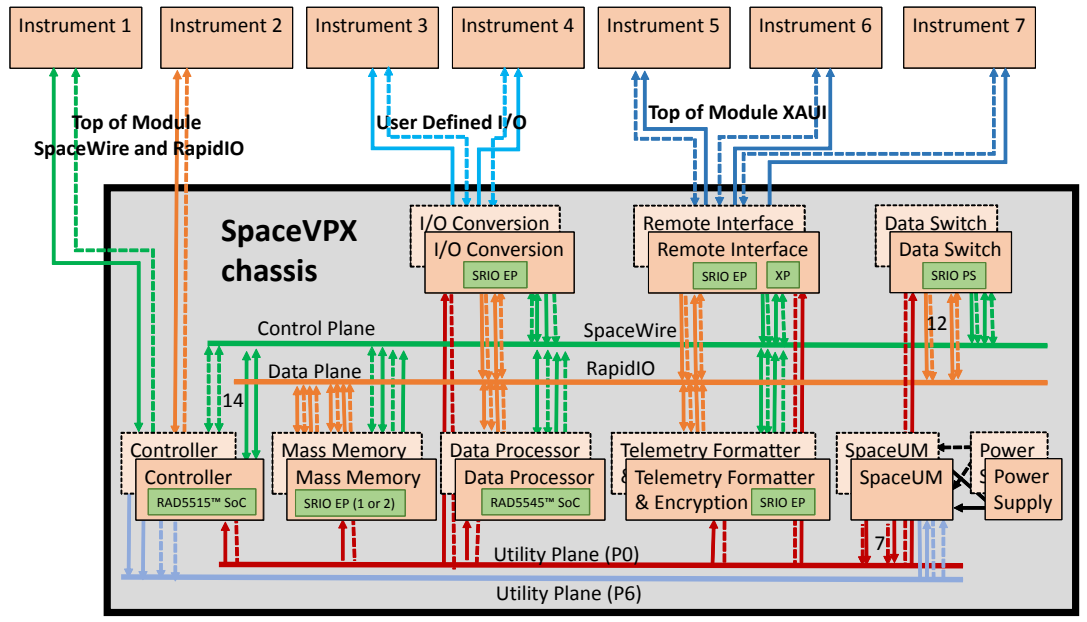


Figure 2: SpaceVPX system using SpaceWire for control and RapidIO for data

BAE Systems has or is developing many of these SpaceVPX modules. This paper will describe the RAD750® single board computer (SBC), which may be used as a small system controller, a data processor or some other intelligent payload. Other modules to be described are the RAD5545™ SBC, which may play similar roles but with up to 10x the performance, a RADSPEED DSP module and a Reconfigurable Computing module, each which may function

as a Data Processor. The green blocks in each diagram highlight developed application specific standard products that provide the major interface and or processing element for that module. Other modules under development but not discussed in this paper include mass memory, data switch, SpaceUM and power supplies. Taken together, systems of various performance and bandwidth can be constructed out of these interoperable modules.

III. RAD750® SBC

The RAD750 SBC, as shown in Figure 3, is based upon the upgrades to BAE Systems' RAD750 CPU. It is our first SpaceVPX module and utilizes the 6U-220 form factor to provide room for 128 MB of radiation-hardened SRAM with SECDED and 512 MB of TMR flash. The RADNET™ SpW-RB4 [10] provides the system interfaces to a RAD750 V2 or V3 processor with 1 MB of L2 cache attached. Four 320 MHz SpaceWire ports connected through an internal nine port router provide over a Gbps of external bandwidth. A 32 bit PCI bus interface is also brought to the SpaceVPX P5/J5 backplane segment for connection to heritage modules over the expansion plane. A dual redundant MIL-STD-1553B interface is implemented and routed on the backplane over user defined pins. A utility FPGA, implemented using the Microsemi RTAX2000S device, provides SpaceVPX connections to the utility plane, conversion of voltage levels to the SRAM and the TMR flash interface. This module can be used as a SpaceVPX controller for systems of up to five modules or with a separate control plane switch module, as many as are needed. It also may be used as medium performance data processor where processing up to 500 DMIPs are required or as a bridge to heritage CompactPCI modules.

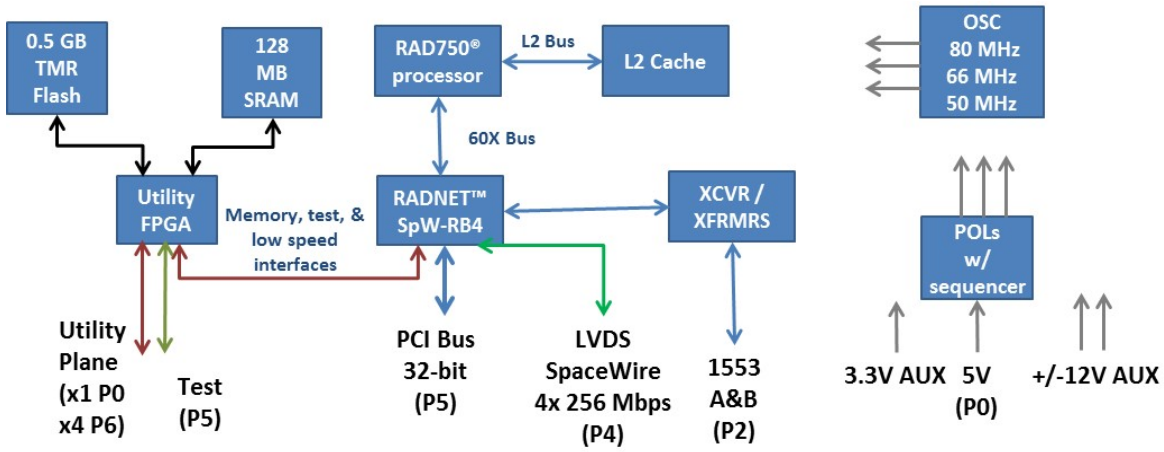


Figure 3: RAD750 SBC block diagram

IV. RAD5545™ SBC

The RAD5545 single board computer (SBC) as shown in Figure 4 is based upon the RAD55xx™ Power Architecture® system-on-chip (SoC) processor platform component that can be configured into multiple personalities including the RAD5545 quad-core processor [5]. The 6U-220 format SBC with SpaceVPX connectors [1] can be used in either a payload or controller slot as defined by the VITA 78 standard leveraging the 16-port SpaceWire router integrated into the RAD55xx platform as the central hub of the SpaceVPX control plane. Twelve of the sixteen SpaceWire links are connected to the SpaceVPX connector, along with three and optionally four of the four lane RapidIO ports, two Gb Ethernet SGMII ports, and the UARTs through external RS422 drivers and receivers.

The other key component on the RAD5545 SBC is also a Microsemi RTAX2000S FPGA. The FPGA includes logic to provide a triple modular redundant (TMR) NAND Flash memory interface driven by the single Flash memory interface on the RAD5545 processor, the signals required for the SpaceVPX utility plane based on I2C interfaces from the RAD5545 processor, an interface to the configuration EEPROM that is fed into the RAD5545 SRAM/EEPROM port, and a 1 pulse-per-second (1 PPS) input/output.

The DDR3 DRAM memory controller of the RAD5545 processor feeds to a ruggedized dual inline memory module (XR DIMM) connector, providing the flexibility to insert RAM modules of different sizes and features onto the base 6U-220 module. The SBC will accept 2, 4 or 8 GB of DDR3 SDRAM with error SECDED correction on a single memory rank using standard commercial DRAM components.

The front side of the card as shown in Figure 5 includes six point-of-load (POL) regulators to generate the various low voltages required that cannot be viably distributed from a centralized power supply. The power up/down sequence is controlled by logic on the back side of the card.

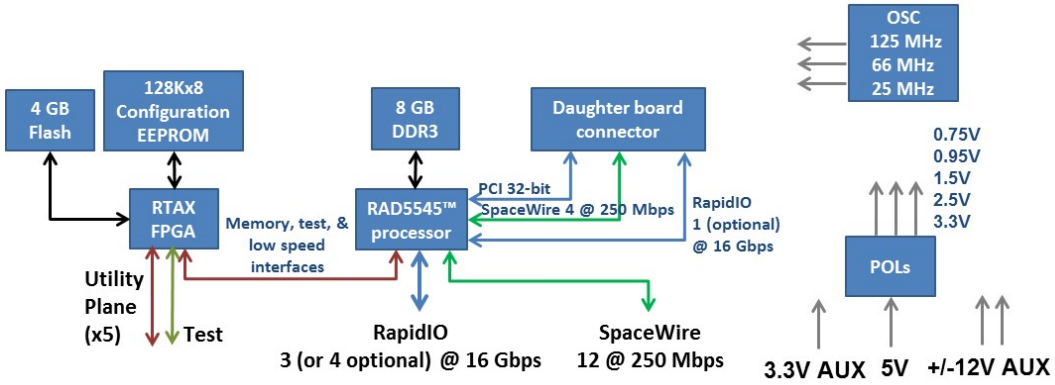


Figure 4: RAD5545 SBC block diagram

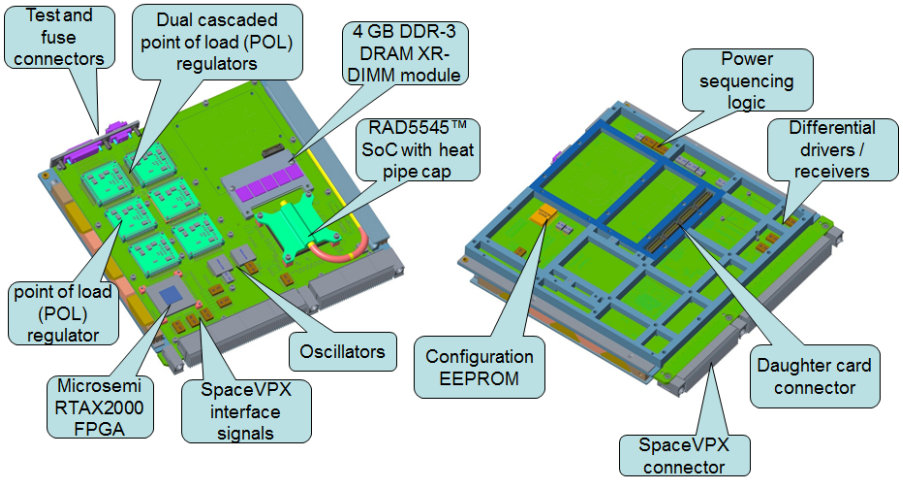


Figure 5: Front and back side CAD drawings of the RAD5545 SBC

The back side of the base module includes the connector for a daughter card 5.22" in length and 3.88" in width. The daughter card may be configured by user to add customized features to the SBC. From the processor, four SpaceWire ports, a 32-bit parallel PCI bus, and optionally one x4 RapidIO port (mutually exclusive with the fourth RapidIO port to the backplane) are routed to the daughter card connector. Power conversion from the base card POL regulators is also supplied through the daughter card connector. Likely applications for the daughter card would be to add additional unique functions or heritage interfaces that would leave the subsystem such as MIL-STD-1553 and Controller Area Network (CAN), both of which employ unique physical layers. There is sufficient space on the daughter card format for one FPGA device.

V. Reconfigurable Computing Module

Many payload electronics boxes require signal processing. The Reconfigurable Computing Module (RCM) meets this need in a SpaceVPX 6U-220 format. A block diagram of the module is shown in Figure 6. Two Xilinx Virtex-5QV RAM-programmable FPGAs provide significant reprogrammable logic for implementing signal processing algorithms. Each FPGA has 1 GB of DDR2 SDRAM attached to it and each connects to a small daughter card (3.8" x 2.86") for personalizing any external I/O that is connected (e.g. ADCs for analog inputs or DACs for analog outputs).

The 18 SERDES lanes on each FPGA are connected to the backplane (8 lanes) through a cross-point switch, to the daughter card (4 lanes) and to the other FPGA (6 lanes). A utility FPGA provides the utility plane connections for the module, a flash interface to up to 8 GB of TMR flash and either a SpaceWire control plane interface or an internal embedded microcontroller. As such the module may be controlled remotely through the SpaceWire port or by loading and executing code on the module. FPGA bit files are stored in the flash and are loaded under direction of the remote or local controller.

Figure 7 shows CAD drawings of the RCM with major components identified. Note that heat pipes are used to remove heat from the two FPGAs to the wedge locks. The two

daughter cards may be plugged into identical connectors on the top of the front view of the module. In order to maximize function, the daughter card may span the two sets of connectors, especially if only one side of the card has implemented function or interfaces. Besides the obvious ADC and DAC functions, FPGA daughter cards may be used for redundant SERDES connections, electrical to optical or extended memories.

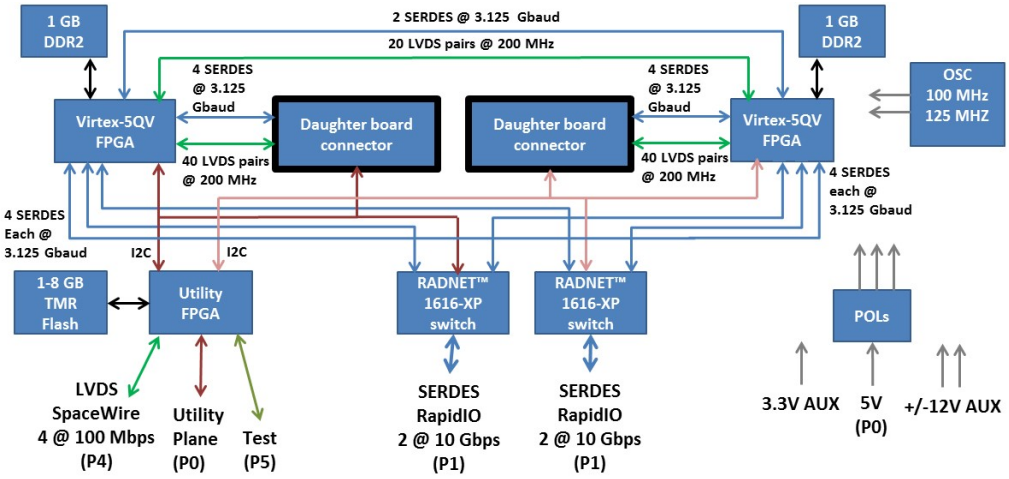


Figure 6: Reconfigurable Computer Module block diagram

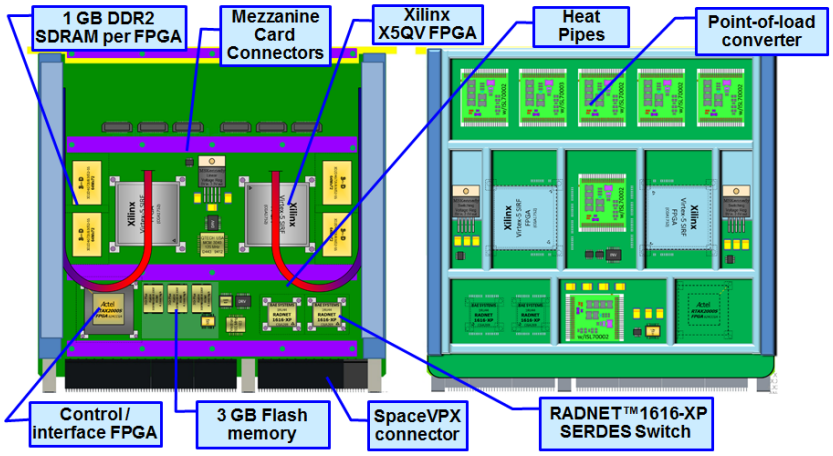


Figure 7: Front and back side CAD drawings of the RCM

VI. RADSPEED™ DSP MODULE

Providing a lower power per GFLOPS signaling processing solution, the RADSPEED DSP SpaceVPX module includes either one or two single instruction, multiple data (SIMD) digital signal processor components, supported by a unique variant of the RAD55xx™ family called the RADSPEED HB [5] [6]. The RADSPEED HB is a host/bridge that communicates with the two DSPs across a unique high performance parallel bus called the ClearConnect® Bridge, shown in the block diagram in Figure 8 as "CCBR".

Each RADSPEED DSP contains two independent multiple thread array processors (MTAP) each of which consists of 76 identical processing elements. Each processing element includes a full double precision floating point engine, fixed point logic unit, multi-port register file, and 6 KB of SRAM. The aggregate capability of both MTAPs is 70 GFLOPS of peak throughput providing up to 140 GFLOPs in total on the module and supplemented by the horsepower of the quad-core

RADSPEED HB as appropriate. Each MTAP includes a dedicated DDR2 DRAM port and the module includes 1 GB of DRAM on each port.

.

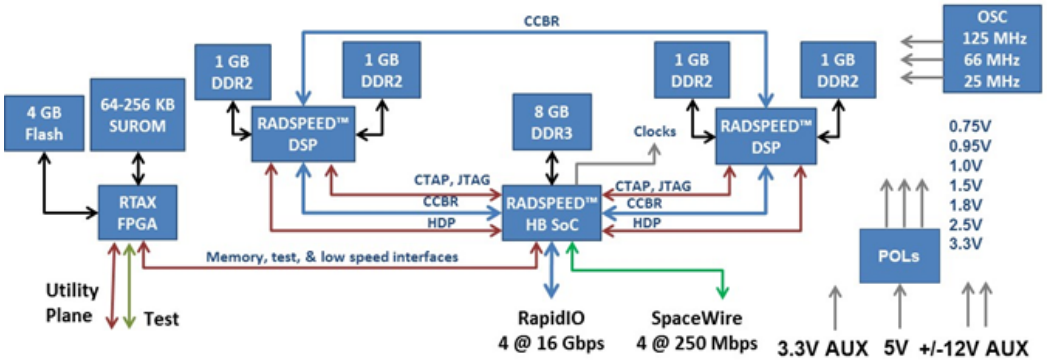


Figure 8: Block diagram of the RADSPEED DSP card

The DDR3 DRAM interface on the RADSPEED HB SoC can support up to 8 GB of memory, mounted on an XR-DIMM identical to the one provided for the RAD5545 SBC. Also, as included on the RAD5545 SBC, an RTAX2000S FPGA is included to provide the triplicated Flash interface and to generate the SpaceVPX Utility plane signals.

Because the process technologies of these key components are different, the supply voltages are also not identical. As a result, the RADSPEED DSP module includes a myriad of POL regulators to generate all of the local voltages. These various voltages are also power sequenced.

The SIMD architecture RADSPEED DSP is well suited to both signal processing and image processing, leveraging features such as an optimized Fast Fourier Transform (FFT) library function. Performance analysis of both Space-Time Adaptive Processing (STAP) and Synthetic Aperture Radar (SAR) functions as well as analysis of hyperspectral imaging have been performed. Some of the benchmarks executed on the RADSPEED DSP are the Complex Ambiguity Function (CAF) [11] and image processing algorithms such as the Harris Corner Detector for feature detection and the Histogram of Oriented Gradient (HOG) for object detection [12]

VII. Module Testing

The RAD750 SBC has completed checkout and is now supporting its applications. The RADSPEED DSP module is in design. The RAD5545 SBC and the RCM should complete fabrication and move to the lab checkout and bring-up by 4Q16. All of these modules align with the SpaceVPX 6U Payload/Controller slot profile family and the latter three have user defined signals in the same locations, enabling them to be interoperable. (The RAD750 SBC with its heritage PCI Bus and 1553 I/Fs on the backplane requires a different slot though user I/O are consistent with the other three modules.) Thus a common test structure and two test backplanes (peripheral and payload) have been created for use across these modules and many others that may be created in the future. The test backplane is designed to route signals for the utility, control, data, and expansion planes for Controller and Payload slot types. Each Module is tested in accordance to their system slot type. Each slot type has a different backplane signal routing. Table 2 captures the different slot types supported for each module described above.

Table 2 – Module Slot Types

Slot Type	Backplane	Module
Controller/Payload /Peripheral	Peripheral	RAD750 SBC with PCI
Controller/Payload	Payload	RAD750 SBC without PCI*
Controller/Payload	Payload	RAD5545 SBC without PCI
Controller/Payload /Peripheral	Peripheral	RAD5545 SBC with PCI*
Payload	Payload	RCM
Peripheral	Payload	RCM with PCI*
Payload	Payload	Single RADSPEED DSP*
Payload	Payload	Dual RADSPEED DSP
Payload	Payload	Storage Module*
Switch	Switch*	RapidIO Packet Switch*
Controller/Switch	Switch*	Controller w/ Packet Switch*

*Future

The test fixture is side loading which allows for backplane probing. The backplane is designed to be compatible with a commercial chassis, leveraging OpenVPX infrastructure and commercial modules for workbench testing. The test fixture supports up to two redundant modules of each type. This will allow for testing of module redundancy. Backplane slot profiles adhere to the SpaceVPX defined profiles as summarized in the family slot profile shown in Figure 9. The main difference between the Peripheral and Payload backplanes are the expansion planes in P2/J2 and P5/J5. The peripheral backplane routes connector single-ended signals for the PCI Bus in P5/J5 and most test user defined signals on P2/J2. The Payload P2/J2 has very few test user defined signals due to routing challenges with the data plane in P1/J1 and instead uses connector differential for these signals on P5/J5. The test interfaces include JTAG connections. Test signals are all common on the backplane regardless of slot profile used and are located in the Utility Plane per the standard.

Figure 11 depicts the tester backplane connections and supported signals required for test of up to two controller or payload modules. Note that the controller cards are tested independently of each other therefore two cards are not required for bring up and test. Single string processing is also supported. The tester backplane routes signals needed for module testing. All interfaces, such as the SpaceWire, RapidIO, clocks and discretes are routed to backplane connectors for testing or wrapping. Management signals on the utility plane are also routed to the backplane.

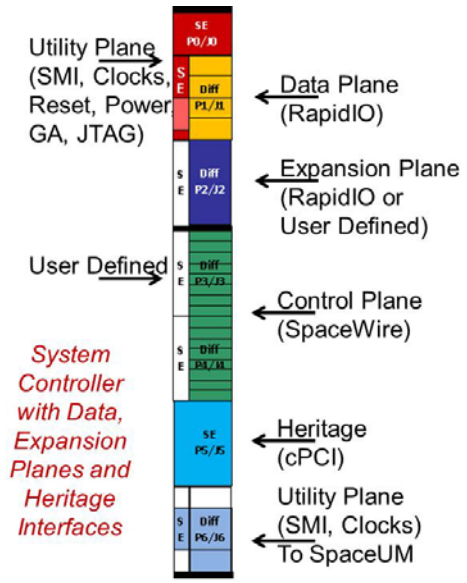


Figure 9: Interface Planes Mapped to Slot Profiles

High performance interfaces such as SpaceWire or RapidIO require test equipment that can drive or monitor these interfaces. Figure 10 diagrams the test setup and general connections for these advanced high performance modules (VPX modules). Existing SpaceWire test equipment such as those from Star Dundee and / or 4 Links will be used for working with and analyzing the SpaceWire signals. RapidFET designed by Fabric Embedded Tool Corporation will be used to test RapidIO interfaces. Because of its diagnostics capability, this tool can be used during board bring up and component testing as well.

Ethernet is an interface that many organizations use in non-space labs for testing or development between terrestrial equipment. Currently being developed with the NSF Center for High-Performance Reconfigurable Computing (CHREC) [13] is an Ethernet - SpaceWire box which will use SpaceWire as a medium to transport Internet protocol (IP) packets. Device drivers will enable these packets to be used directly by processors in the SpaceVPX module with IP awareness. This will facilitate lab testing and application development without requiring an Ethernet port on the space hardware. The system test equipment (STE) PC can communicate with such processor applications as if they are IP. Similarly, IP will be usable for inter-processor communication over the SpaceWire network within the box.

Also being developed is a software/hardware Prototype development fixture which will allow customers to develop applications on these modules without requiring the System Tester. This will expedite customer development.

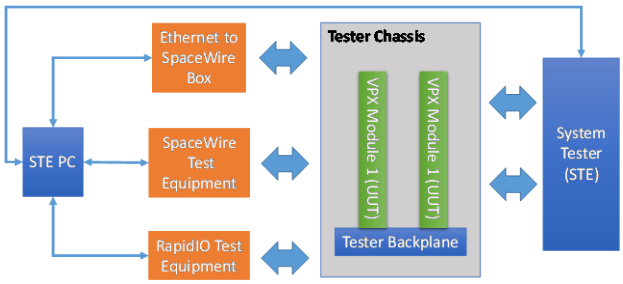


Figure 10: Test Setup Block Diagram

VIII. Summary

This paper has described a family of SpaceVPX modules that will greatly increase the processing options for creating scalable single string and single point fault tolerant payloads. All of the modules utilize the same group of SpaceVPX slot profiles and thus are interoperable with appropriate backplane routings of the utility, control, data and expansion planes. SpaceWire is part of all of these and provides a critical function as the control plane as well as medium speed data movement. Several other modules are under development that will provide additional functions and connectivity within the same structures.

Table 2 provides a summary of the modules presented along with their key technical specifications.

References

[1] "SpaceVPX Standard", ANSI/VITA 78.00-2015, www.vita.com

[2] Collier, Charles Patrick, et al., "Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space", Proceedings of 2015 IEEE Aerospace Conference, Big Sky MT USA, March 2015

[3] P. Collier, J. Marshall, R. Berger, M. Enoch, S. Goedeke, "Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space", AIAA 8 Reinventing Space 2013 Conference Proceedings, Los Angeles CA USA, September 2013.

[4] D. Rickard, et. al., "On-Board Networks with Radiation Hardened 45nm SOI Standard Components", Proceedings of the IEEE Aerospace 2015 Conference, Big Sky MT USA, March 2015.

[5] R. Berger, et. al., "Quad-Core Radiation-Hardened System-on-Chip Power Architecture Processor", Proceedings of the IEEE Aerospace 2015 Conference, Big Sky MT USA, March 2015.

[6] J. Marshall and R. Berger, "High Performance Network Components for Scalable Spaceborne Processing Needs", Proceedings of the 2016 International SpaceWire Conference, Yokohama, Japan, October 2016.

[7] R. Berger, et al, "The RAD750 – A Radiation Hardened PowerPC Processor for High Performance Spaceborne

Applications", IEEE Aerospace Conference 2001, Big Sky MT, USA, March 2001

[8] RapidIO Interconnect Specification 3.1, www.rapidio.org, September 2014

[9] ECSS Standard ECSS-E-ST-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, July 2008.

[10] J. Marshall, D. Stanley, J. Robertson, "Matching Processor Performance to Mission Application Needs", Infotech@Aerospace 2011 Conference Proceedings, St. Louis MO, USA, March 2011.

[11] J. Marshall, et. Al., "Applying a High Performance Tiled Rad-Hard Digital Signal Processor to Spaceborne Applications", IEEE Aerospace Conference 2012, Big Sky MT, USA, March 2012.

[12] Fijany, A. and Hosseini, F., "Image processing applications on a low power highly parallel SIMD architecture", IEEE Aerospace Conference 2011", Big Sky MT, USA, March 2011.

[13] www.CHREC.org

All figures are Copyright BAE Systems – used with permission

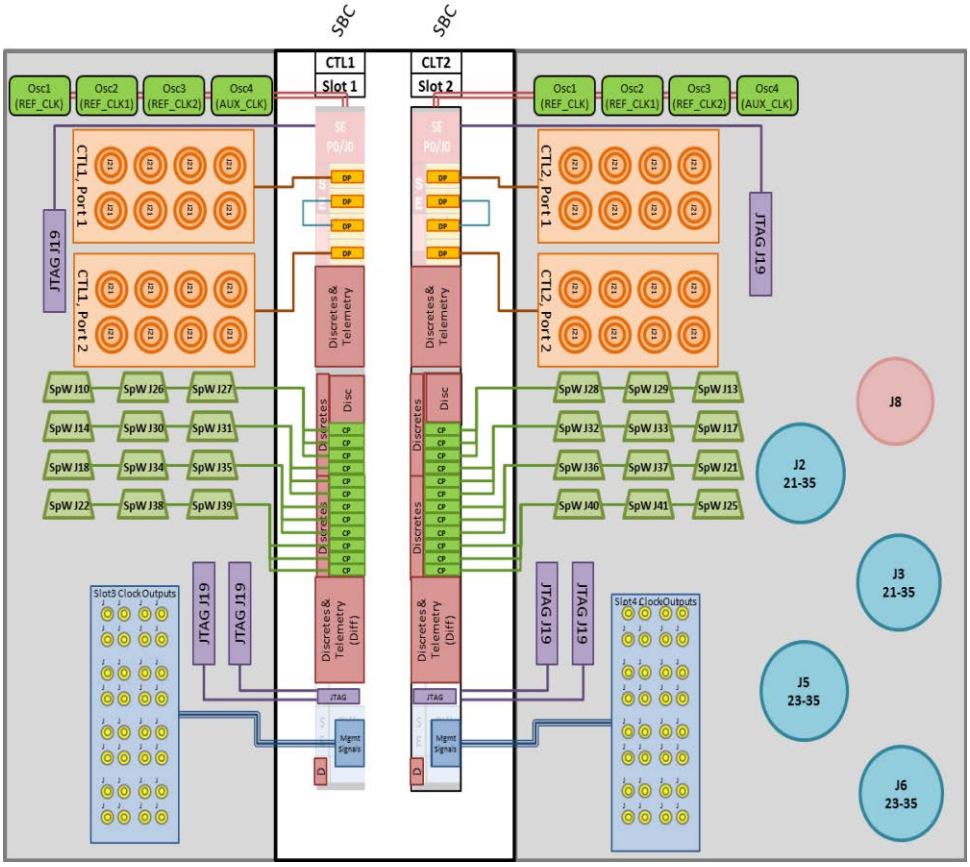


Figure 11: Controller/Payload Tester Backplane. (SpW = SpaceWire)

Table 2 – High Performance SpaceVPX Module using SpaceWire Summary

Module	MIPS	MFLOPS	GB RAM	GB NVM	Data Port Rate Gbps	Data Ports	SpaceWire Port Data Rate Mbps	Space-Wire Ports	Daughter Board Connectors
RAD750 SBC	500	250	0.128	0.5	1	1	256	4	
RAD5545 SBC	5600	3700	8	4	16	4	320	12	SBC, DIMM
Dual V5 FPGA RCC		10000	2	4	10	6	100	4	2-RCC
Dual 90nm RADSPEED DSP	6200	143700	12	4	16	4	320	4	

Test & Verification (Short)

MOST: Modeling of SpaceWire & SpaceFibre traffic

NOT PERMITTED TO PUBLISH PAPER

SpaceFibre (Short)

SpaceFibre and Serial RapidIO Network Layers (GRSPFI/GRSRIO)

SpaceFibre, Short Paper

Felix Siegle, Sandi Habinc
Cobham Gaisler AB
Gothenburg, Sweden
[felix | sandi] @gaisler.com

Kostas Marinis
European Space Agency
Noordwijk, The Netherlands
kostas.marinis@esa.int

***Abstract*—Cobham Gaisler presents a network layer implementation for easy integration of SpaceFibre and Serial RapidIO into modern System-on-Chips.**

***Index Terms*— Serial RapidIO, SpaceFibre, System-on-Chips**

I. Introduction

SpaceFibre (SpFi) [1] is a new high-speed serial data link specifically designed for spaceflight applications that incorporates several Quality-of-Service (QoS) techniques. Independent communication channels can be combined into a single network stream by means of virtual channels. The virtual channels are multiplexed based on reserved bandwidth, priorities, time-slots, or a combination of these mechanisms. Integrated Fault Detection, Isolation and Recovery (FDIR) support guarantees fault-free communication. SpFi is particularly well suited for, but not limited to, device-to-device intercommunication.

Serial RapidIO (SRIO) [2] is another high-speed serial data link considered for space applications. With support for inter-process messages, doorbell messages and memory I/O operations, it is particularly well suited for chip-to-chip and board-to-board intercommunication.

Cobham Gaisler is currently working on a modern network layer implementation for both SpaceFibre and Serial RapidIO (also referred to as "logical layer"). Due to major similarities between these two communication standards, a single, modular concept is developed, in which only a few blocks must be swapped depending on the underlying physical layer. The new architecture advances earlier concepts as it is specifically designed to handle high data throughput rates and also targets multi-core and multi-memory systems.

II. Differences and Similarities between SpaceFibre and Serial RapidIO

While there are fundamental differences between SpaceFibre and Serial RapidIO in terms of error-recovery management and QoS mechanisms, the actual data transmitted over both protocols shows many similarities.

SpaceFibre uses for raw data transmission the simple SpaceWire packet format [3], which consists of one or several addresses, the packet payload and an End-of-Packet (EOP) marker. In theory, the packet length of such a raw packet is not limited. By using the additional Remote Memory Access Protocol (RMAP) protocol [4], remote memory I/O access becomes possible via SpaceWire/SpaceFibre as well. SpaceFibre also supports broadcast frames, which are multi-purpose high-priority messages. These messages are comparable to SpaceWire time-codes but in addition to a simple sequence number they also comprise a data payload of 8 bytes.

Serial RapidIO is a more packet-based protocol that already includes remote memory I/O access operations (defined in the logical I/O specification) and messages (defined in the messaging specification). Data streaming is in contrast to SpaceFibre rather a supplement than an essential part, which also reflects the different application domains of these two protocols since SRIO is clearly targeted at chip-to-chip communication. However, with the optional data streaming protocol (defined in the data streaming logical specification), complex streaming applications can also be implemented over Serial RapidIO.

Thus, by adding RMAP to SpaceFibre and the data streaming protocol to Serial RapidIO, three basic data transfer types can be identified that must be handled by a common network layer, see also Table I. The first type describes interrupt-style short messages, which are used for example for processor to processor communication. The SpFi broadcast messages and the SRIO doorbell messages clearly fall into this category. The SRIO data message, however, is more complex as it can also carry larger payloads of up to 4K data and it therefore does not have a real equivalent in the SpaceFibre domain. The second type describes memory I/O access operations, for example read, write or read-modify-write operations and for this data transfer type the differences between RMAP and the I/O logical specification are rather small. The third type describes data streams, which are made possible over SRIO using the data streaming protocol. In principle, streaming over SpaceFibre can be realized by only using the SpaceWire packet format but for

more advanced applications it might be worth to consider an additional protocol like the CCSDS packet transfer protocol [5].

TABLE I. SUPPORTED DATA TRANSFER TYPES

Type	SpaceFibre	Serial RapidIO
Interrupt messages	Broadcasts	Messages, Doorbells
Memory I/O access	RMAP Protocol	I/O Logical Spec.
Data Streams	SpaceWire Packets and/or CCSDS packet transfer protocol	Data Streaming Specification

III. NETWORK LAYER ARCHITECTURE

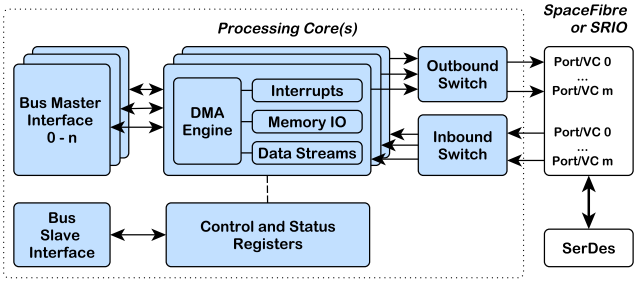


Fig. 1. Simplified block diagram of the GRSPFI/GRSRIO Network Layer

A. Overview

A simplified block diagram of the network layer is shown in Figure 1. The virtual channels of the SpaceFibre port or several ports of a Serial RapidIO endpoint are connected to one or multiple network layer processing cores via inbound and outbound routing switches. Each processing core is connected to a generic bus master. At present, the AMBA bus is directly supported for a simple integration with the Gaisler product line.

A processing core comprises a configurable number of transmission and reception queues for interrupt messages, one transmission and one reception queue for data streams and one transmission queue for memory I/O access operations. Control and status registers for each queue are clearly separated and can be mapped for example to 4K boundaries to allow a Memory Management Unit (MMU) to restrict the access of specific queues to specific CPU cores.

In a multi-memory system, several processing cores can be instantiated, which all comprise their own bus master. Then, data can concurrently be transmitted and received over several SpFi virtual channels / SRIO ports at the same time.

The front-end of the bus master, i.e. the interface to the processor core, has separate FIFO-like read- and write-channels. The back-end of the bus master can be configured for different bus widths (e.g. 32 bit, 64 bit or 128 bit) and bus protocols (AMBA or AXI4). Due to the generic FIFO interface architecture, the bus master could even be replaced by custom-built logic, e.g. by a bridge to a Network-on-Chip (NoC), a local on-chip memory or a direct interface to a microcontroller.

This highly flexible design allows many different System-on-Chip (SoC) approaches. For instance, several SpFi virtual channels can be shared by one single core, multiple cores can manage one single SpFi virtual channel, or specific virtual channels can exclusively be managed by a particular CPU core.

B. Handling of interrupt messages

1) Data Messages (SRIO)

For these messages, multiple queues on both inbound and outbound side can be installed during design time by corresponding VHDL generics. If multiple processing cores are used, a particular queue is always managed by one particular processing core, although this affiliation can be reprogrammed dynamically during runtime by software.

On inbound side, each message reception queue comprises a filter that allows the mapping of a mailbox number or a range of mailbox numbers to this specific queue. For instance, the user could set up five queues, the first four queues accepting messages addressed to mailbox 0 to 3 and the fifth queue accepting messages addressed to all other, higher mailbox numbers. As an additional feature, data messages can also be filtered based on their destination ID. If a message is received, which is addressed to a mailbox number / destination ID combination that is not accepted by any message reception queue, the processing core automatically generates and transmits ERROR responses to the source node. Concurrent reception of letters is not supported. If a message packet is received with a letter number different to the one of the currently processed message, this message is dropped and RETRY responses are generated and transmitted to the source node. Both aforementioned events are logged in status register fields and can optionally trigger interrupts.

On outbound side, each message transmission queue is set up to transmit messages to one particular (or several alternative) port numbers. However, this port number(s) can also be changed dynamically during runtime by software. The queues are serviced by the processing core in a round-robin fashion.

Each message reception and transmission queue comprises a circular buffer that contains DMA descriptors set up by software. The depth of the circular buffer as well as its location in memory can be changed during runtime, effectively allowing the software to swap queues if necessary. On both inbound and outbound side, the head pointer to the circular buffer is managed by software and the tail pointer is managed by the DMA engine.

For incoming messages, the software first sets up descriptors that point to free memory spaces and then increments the head pointer accordingly. By doing so, it signals to the DMA engine that the new descriptors are ready to be processed. Once the DMA engine has received and stored a message successfully, the current descriptor is updated and the tail pointer is incremented. By doing so, the descriptor and thus also the received message is handed over to software. Furthermore, the processing core generates and transmits a DONE response to the source node. Each message descriptor is updated with a time-stamp value after successful reception that is taken from an external source, e.g. an external timer module.

For outgoing messages, the descriptors point to memory spaces containing the message payloads to be transmitted. Again, the software signals to the DMA engine that the

descriptors are ready for processing by incrementing the head pointer. Once the DMA engine (i) committed a message successfully to the Serial Rapid IO endpoint and (ii) received a response (DONE, ERROR) from the destination node, the descriptor is updated with the result and the tail pointer is incremented accordingly. By doing so, the software is informed that the descriptors are free again and both the descriptors and the corresponding memory spaces can be reused by software. In case of a RETRY response, the processing core retries the message automatically until the message is either accepted by the destination node or until a retry threshold level is reached. Not until then, the descriptor is handed over to software.

For both the outbound and inbound side, interrupts can be enabled for each message independently that are either triggered after successful processing or when an error condition occurred. In addition, an interrupt can be enabled on inbound side that is triggered when a particular message reception queue becomes full. In case of a full message reception queue, the processing core also generates and transmits RETRY responses to the source node.

For multi-packet messages, timeout mechanisms are available on both outbound (request-to-response) and inbound side (response-to-request) that limit the allowed time in which the destination node sends a response packet (outbound side) or the next message segment (inbound side). Timeouts are flagged in status registers and can also be set up to trigger interrupts.

2) Doorbell Messages (SRIO) / Broadcast Messages (SpFi)

For doorbell/broadcast messages, multiple circular buffers exist for both inbound messages and outbound messages. The buffers use the same concept of handshake between software and hardware as is the case for other messages, that is, the software manages a head pointer and the DMA engine manages a tail pointer. In contrast to normal messages, however, the buffers do not contain descriptors but rather the doorbell/ broadcast messages themselves.

On inbound side, the DMA engine stores incoming doorbell/broadcast messages automatically to an assigned buffer as long as free space is available, that is, as long as the head pointer is ahead of the tail pointer. The doorbell/broadcast messages can be assigned to particular reception buffers by filtering their destination ID (SRIO) or their broadcast channel number or a range of broadcast channel numbers (SpFi). Once a message is successfully received and stored to memory, the DMA engine increments the tail pointer accordingly and the processing core generates and transmits a DONE response to the source node (only SRIO, SpFi broadcasts are never acknowledged). Under normal conditions the software will ensure that the head pointer of the doorbell reception queue is ahead of the tail pointer by a couple of messages. However, if the software cannot process the incoming doorbell/broadcast messages fast enough, it can apply back-pressure by simply not incrementing the head pointer. If the reception buffer is full, the processing core automatically generates and transmits RETRY responses to the source node (only SRIO). Each doorbell/broadcast message is stored with a time-stamp value that is taken from an external source, e.g. an external timer module.

On outbound side, the software can set up one or several doorbell/broadcast messages at once and then increment the head pointer as desired. Once the DMA engine committed a doorbell/broadcast message successfully to the SRIO/SpFi endpoint the tail pointer is incremented. In case of a RETRY response, the processing core retries the doorbell message automatically until the message is either accepted by the destination node or until a retry threshold level is reached (only SRIO). Not until then, the buffer space is handed over to software.

For both the outbound and inbound side, interrupts can be enabled that are either triggered after successful processing or when an error condition occurred. In addition, an interrupt can be enabled on inbound side that is triggered when the doorbell/broadcast message reception buffer becomes full.

Furthermore, one doorbell/broadcast output signal for each processing core is available at the network layer port. If enabled, the signal is pulsed after the reception of a doorbell/broadcast message and the doorbell/broadcast message payload value is signaled on dedicated output pins. Optionally, the payload of the doorbell/broadcast is first compared to a programmable value and mask. Only if the comparison succeeds, the output signal is pulsed. In addition, doorbell/broadcast input signals for each processing core enable external hardware components to generate and transmit doorbell/broadcast messages directly. The payload value of the doorbell/broadcast message is also provided by the external component via dedicated input signals.

C. Handling of memory I/O operations (SRIO/SpFi)

On outbound side, I/O operations are managed in a similar way as outbound messages. Each processing core comprises one I/O transmission queue. This queue makes use of a circular buffer that contains DMA descriptors set up by software. The depth of the circular buffer as well as its location in memory can be changed during runtime, effectively allowing the software to swap queues if necessary. The head pointer to the circular buffer is managed by software and the tail pointer is managed by the DMA engine.

For outgoing I/O operations, the descriptors contain all required information about the operation itself as well as pointers to memory spaces, which are either reserved for data that is read from a remote device or which contain data that shall be written to a remote device.

Just as for the message handling, the software signals to the DMA engine that the descriptors are ready for processing by incrementing the head pointer. Once the DMA engine (i) committed an I/O operation successfully to the SRIO/SpFi endpoint and (ii) received a response from the destination node, the descriptor is updated with the result and the tail pointer is incremented accordingly. By doing so, the DMA engine hands over the descriptor as well as the memory space to the software for further processing.

Incoming I/O operations can gain direct access to the local memory space. Translation between SRIO/RMAP addresses and local addresses is accomplished by an optional, fixed memory offset value. Memory protection can be implemented

by defining up to four memory partitions where each partition can be configured to be either read and writable or read-only. Interrupts can be enabled for successful memory accesses. Furthermore, interrupts can be generated in case of errors on the bus or if the remote node tries to access a forbidden memory area. Then, the processing core also generates and transmits an ERROR response to the source node.

D. Handling of data streams

The handling of data streams is not yet fully defined but it is planned that raw SpaceWire packets and CCSDS packets will be supported for SpaceFibre and the data streaming protocol for Serial RapidIO. On inbound and outbound side, data streams will be managed in a similar way as messages. Each processing core will comprise multiple transmission and reception queues, which store DMA descriptors in circular buffer structures.

IV. Verification

At present, a preliminary version of the GRSRIO Serial RapidIO network layer has been implemented that includes one processing core and that implements all data transfer types except of the optional data stream protocol.

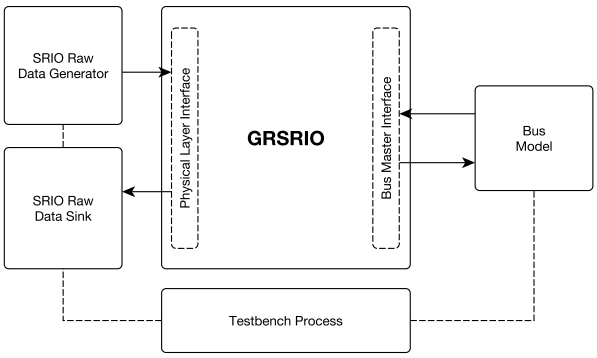


Fig. 2. Simplified block diagram of the GRSRIO testbench environment

A full VHDL testbench environment has been set up, see Figure 2 for a block diagram. The GRSRIO IP core is connected to two concurrent processes mimicking a Serial RapidIO physical layer and another process simulating the AMBA bus master. The main testbench process runs 38 tests altogether, covering all aspects of the logical I/O and messaging specification and is achieving 100% statement coverage.

V. Implementation Results

Example synthesis results for a GRSRIO IP core with two transmission and reception queues for messages and two transmission and reception buffers for doorbell messages can be found in Table II for a Xilinx Virtex-5 FX130 device. The internal data path of the GRSRIO IP core is 128-bit wide.

TABLE II. Implementation Results on Virtex-5 FX130

Max. Throughput Rate:	> 25 Gbps (f = 156.25 MHz)
Slice LUTs:	7619/81920 (9%)
Slice Flip-flops:	2218/81920 (2%)
Block RAMs:	2/298 (0%)

VI. Conclusions

Cobham Gaisler offers with the SpaceFibre and Serial RapidIO network layers innovative solutions for the integration of these protocols into modern System-on-Chips with multiple cores and/or multiple memories. The flexible architecture offers good scalability and can support several bus back-ends like AMBA and AXI4. A preliminary version of the Serial RapidIO network layer was fully verified and synthesis results show high maximum throughput rates at reasonable resource utilization.

Acknowledgment

The development of the Serial RapidIO logical layer is funded by the European Space Agency as part of the “Inter-Processor Link for Future OBCs” activity.

References

[1] Space Technology Center/University of Dundee, “SpaceFibre Specification Draft H1”, Aug. 2015.

[2] RapidIO Trade Association, "RapidIO Interconnect Specification. Revision 2.1". Aug. 2009. [Online]. Available: www.rapidio.org/rapidio-specifications/.

[3] European Cooperation for Space Standardization, “SpaceWire – Links, nodes, routers and networks. ECSS-E-ST-50-12C”. July 2008.

[4] European Cooperation for Space Standardization, “SpaceWire – Remote Memory Access Protocol. ECSS-E-ST-50-52C”. Feb. 2010.

[5] European Cooperation for Space Standardization, “SpaceWire – CCSDS Packet Transfer Protocol. ECSS-E-ST-50-53C”. Feb. 2010.

Design and implementation of test equipment for SpaceFibre links

SpaceFibre, Short Paper

Daniele Davalle, Alessandro Leoni, Luca Dello Sterpaio, Luca Fanucci
Dept. of Information Engineering
University of Pisa
Via Caruso 16, 56122, Pisa, Italy
daniele.davalle@for.unipi.it, alessandro.leoni@ing.unipi.it, luca.dellosterpaio@ing.unipi.it, luca.fanucci@unipi.it

***Abstract*—SpaceFibre is the upcoming European standard for on-board high-speed communications. The need for data-rate beyond 1 Gb/s is already present in space missions, and it is currently fulfilled by non-standard approaches based on Serialiser/Deserialiser components such as Texas Instruments TLK2711. The SpaceFibre standard also integrates Quality of Service and Fault Detection, Isolation and Recovery mechanisms, which allow a highly reliable communication, suitable for space systems.**

The abovementioned features make the SpaceFibre standard undoubtedly complex; therefore an adequate test equipment is necessary for the validation of systems based on this standard.

In this paper, a test equipment for SpaceFibre links is presented. This is designed to support the development of new SpaceFibre devices, as well as complex systems based on SpaceFibre. A system demonstrator was implemented to validate the equipment features.

***Index Terms*—SpaceFibre, test equipment, high-speed serial link, EGSE (Electrical Ground Support Equipment)**

I. Introduction

Modern and forthcoming missions for Earth Observation and science (e.g. Euclid, Juice, Metop-SG, CarbonSat…), are more and more demanding very high-speed reliable data transmission especially within the different units in a payload (beyond 2 Gbps) . Thanks to the evolution of technology for detectors, each single payload can comprise different bandwidth/channels of operations at a very high-speed. In such a context, being able to manage the science data handling and transmission within a payload and from payloads to platforms on-board is still an open point. SpaceWire networks are suitable for single links working up to 400 Mb/s while there is still no standardised solution for higher data-rate.

The European Space Agency (ESA) is therefore finalising a new standard for high-speed data links called SpaceFibre [1], supporting data-rates beyond 2 Gb/s. As such, the SpaceFibre standard allows highly reliable and very high speed point-to-point connections. SpaceFibre defines a complex protocol in order to cope with such stringent requirements, therefore specific competences and a considerable effort are necessary to develop and validate systems based on this standard.

In this paper, a device for test and validation of SpaceFibre-based systems is presented, which provides a powerful yet simple way of validating and debugging such systems. Among its features, the SpaceFibre test equipment can be used to analyse SpaceFibre traffic at different protocol levels, inject SpaceFibre packets and verify the conformance to the SpaceFibre standard. The SpaceFibre test equipment can be controlled by means of a PC through a user-friendly Graphical User Interface. The main use cases of the SpaceFibre test equipment and test results are also presented.

The SpaceFibre test equipment is built upon the experience of the University of Pisa on the SpaceFibre standard. Indeed, the University of Pisa has been working on the SpaceFibre standard since 2014, under a collaboration with ESA for the review and consolidation of the SpaceFibre draft standard. During the work on the topic, a SpaceFibre interface IP-core was independently developed and interoperability with other implementations was demonstrated [2]. The design of the IP-core was carried out using a robust hardware design and verification flow, involving the development of a formal verification environment to prove the functionality and compliance of the IP-core to the SpaceFibre standard [3].

This paper is organised as follows:

The current state-of-the-art in the field of test equipment for SpaceFibre is presented in Section II.

Section III presents the proposed SpaceFibre test equipment.

Section IV describes the implementation of such test equipment.

Section V shows the tests on the system prototype.

Finally, the conclusions are drawn in Section VI.

II. Related work

SpaceFibre is a high-speed serial link standard, specifically designed for use on-board spacecraft, and developed by the University of Dundee for ESA. The standard is currently under finalisation by the European Cooperation for Space Standardization (ECSS).

SpaceFibre is able to operate over fibre-optic and copper cable and supports data rates as 2.5 Gb/s, 3.125 Gb/s and 6.25 Gb/s per lane, up to a maximum of 20 Gb/s with multilane design. SpaceFibre is backwards compatible with the SpaceWire

standard at packet level, which allows easy integration of the new standard into existing systems based on SpaceWire.

SpaceFibre will be used in all cases where data-rate requirement is 1 Gb/s and beyond, which is already a requirement for currently designed space mission payloads. Other than the high-data rate, the SpaceFibre link will take advantage of Quality of Service (QoS) and Fault Detection, Isolation and Recovery (FDIR) capabilities. One of the possible applications of SpaceFibre is the multiplexing of multiple SpaceWire channels over a single SpaceFibre link, in order to reduce the harness on the spacecraft [4]. Each SpaceFibre Virtual Channel (VC) can be seen as a SpaceWire link; therefore, a bundle of SpaceWire links can be replaced by a single SpaceFibre link resulting in a considerable mass reduction and compact system setup, adding QoS and FDIR features to SpaceWire links.

STAR-Dundee produced StarFire [5], a test unit to support the development and early adoption of the SpaceFibre standard. The unit can generate random packets over the SpaceFibre links as well as consume received data. StarFire can route the SpaceWire ports to VC0 or VC1 of the SpaceFibre interfaces. The unit supports 2.5 Gb/s single-lane SpaceFibre link rate.

III. System definition

The SpaceFibre test equipment presented in this paper, also referred to as "SpaceFibre Analyser", comprises the following main features:

- Two SpaceFibre interfaces
- Two SpaceWire interfaces
- Compliant with SpaceFibre and SpaceWire standards
- Ethernet / PCIe interfaces for host PC communication
- Real-time communication with the host PC through PCI communication
- 8 Virtual channels for each SpaceFibre interface, which is seen as sufficient for present and future device needs
- SpaceWire/SpaceFibre bridging
- Link-Analyser mode to monitor the SpaceFibre/SpaceWire links
- Can be used as Electrical Ground Support Equipment (EGSE) for the validation of satellites based on SpaceFibre/SpaceWire.
- Hardware packet generator and packet consumer to allow the easy saturation of the SpaceFibre link and enable stress testing of the SpaceFibre network
- Error injection/Word replacement capability to facilitate conformance testing of the system under-test
- TX/RX trace memory of 8192 4-byte words to check protocol-specific features such as flow-control, acknowledgement, frame re-transmission
- Simple to use with either graphical user interface, or command line interface for test automation.

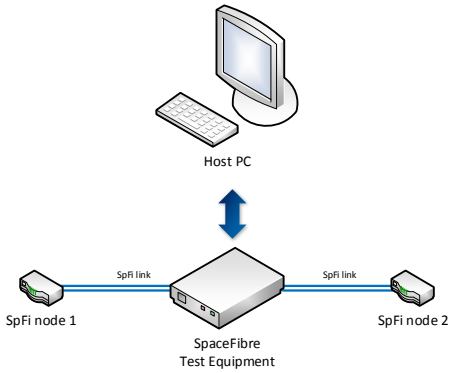


Fig. 1. SpaceFibre test equipment

The test equipment features a sophisticated error injection mechanism on the SpaceFibre link in order to test all the functionality of the Device Under Test. The error injection is available in the following different options:

- *Bit Error Rate (BER) mode*, in which the user sets a desired BER and the test equipment randomly injects bit flips on the TX/RX data according to the selected BER.
- *Bit flip*, in which the user sets the desired word to corrupt
- *Word replacement*, in which a certain word can be replaced by another one selected by the user

In particular, the *bit flip* and *word replacement* modes are useful to test various corner cases of the protocol, e.g., device not receiving ACKs, device not receiving FCTs, etc. All the error injection options listed above, are fully customisable by the user through the Graphical User Interface on the host PC.

In the following sections, the use modes are briefly described.

A. EGSE operation mode

In EGSE operation mode, the SpaceFibre test equipment can be used to emulate a device in a SpaceFibre network, generating predefined packets and responding to user-defined packets. The SpaceFibre test equipment is able to generate and to consume SpaceFibre packets in real time. The user can choose to use the internal hardware packet generator/consumer or reading/writing packet contents from/to a file on the host PC.

B. Link Analyser operation mode

The SpaceFibre test equipment allows the monitoring of the link. In this operation mode the SpaceFibre test equipment is connected in the middle of two different SpaceFibre nodes.

The SpaceFibre traffic that flows on the link can be monitored by the user through the host PC. Errors can be injected on the link to verify the reliability of the communication.

C. Conformance tester operation mode

The SpaceFibre Analyser provides to the user a list of conformance tests, in order to verify the correct features functionality expected from the SpaceFibre standard.

Tests available are useful for the verification of the SpaceFibre initialization protocol, FDIR and QoS capabilities of DUT, and for the correct handling of corner cases such as reception of empty packets.

IV. Implementation

The proposed SpaceFibre Analyser is implemented as a highly-optimized AXI4-centric system. AXI4 is part of the AMBA 4 specification for high performance systems. All the peripherals are connected to the AXI subsystem, this allows to have a high flexibility and, at the same time, to move data among the different interfaces at a very high speed.

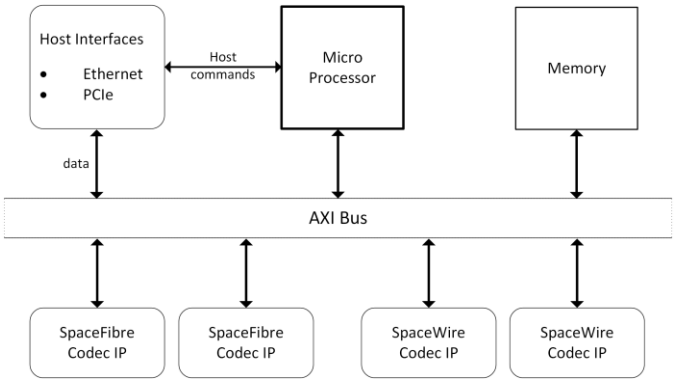


Fig. 2. SpaceFibre test equipment implementation overview

The user, interacting with the GUI, generates commands that are forwarded through the Host interface to the Analyser Microprocessor. The Microprocessor is responsible for the configuration and the control of the entire SpaceFibre test equipment. The Host interface can also be exploited for sending and receiving high-speed data streams from one, or multiple, SpaceFibre and SpaceWire ports.

SpaceFibre and SpaceWire CODEC IPs used are from IngeniArs S.r.l. The IngeniArs SpaceWire CODEC IP-core [6] has considerable heritage in space projects, both on ground and flight hardware. The IngeniArs SpaceFibre CODEC IP-core [7] was intensively verified with a complete and structured SystemVerilog/UVM based test environment reaching the complete code coverage of the RTL [3]. Additionally, the SpaceFibre CODEC IP was demonstrated interoperable with third-party SpaceFibre equipment such as the StarFire equipment [2].

The SpaceFibre and SpaceWire CODEC IPs share a common implementation structure, as depicted in Figure 3. They are equipped with a high-performance *AXI interface* and a highly optimised *DMA engine* interfacing with the AXI system, which allows efficient data movement on the system bus connecting the system memory to the SpaceFibre/SpaceWire ports. At the same time, it allows to realise a simple yet efficient bridging between SpaceFibre and SpaceWire ports.

The *Hardware Packet Generator and Consumer* are implemented in such a way they do not interfere with the other peripherals of the test equipment. They can unobtrusively fulfil the SpaceFibre or SpaceWire link bandwidth.

The Error Injection module acts at the lowest possible word-level, allowing to accurately insert very specific errors on the data stream in order to stimulate complex scenarios.

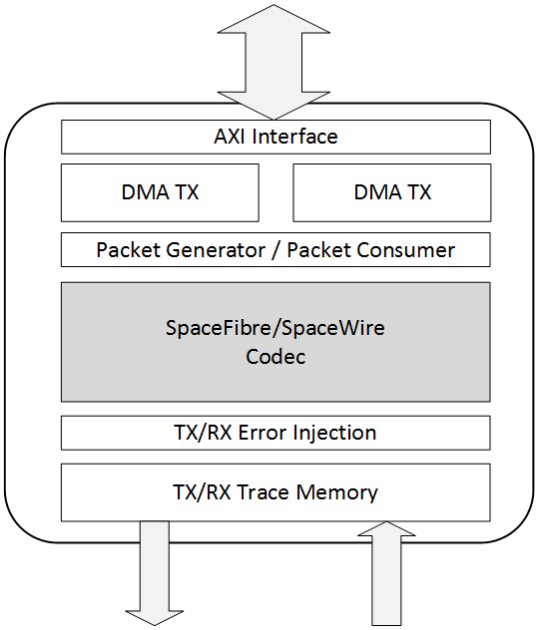


Fig. 3. SpaceFibre/SpaceWire Codec IP structure

The Trace Memory, finally, can be programmed to trigger on a user defined word and show to the user all the words flown through the port within a time window centred on the triggered word.

Each subsystem of a SpaceFibre/SpaceWire Codec IP, and of the Analyser in general, is highly configurable by the user at run time using the GUI, thanks to the specific system architecture adopted.

V. Test and Results

The complete SpaceFibre test equipment was tested in with the two SpaceFibre ports connected with a physical loopback, as shown in Figure 4.

The SpaceFibre test equipment is connected to the Host PC running the GUI through the Ethernet port and the loopback on SpaceFibre ports is realised through an eSATA cable.

Many specific tests were carried out in order to verify the correct behaviour of all the subsystems.

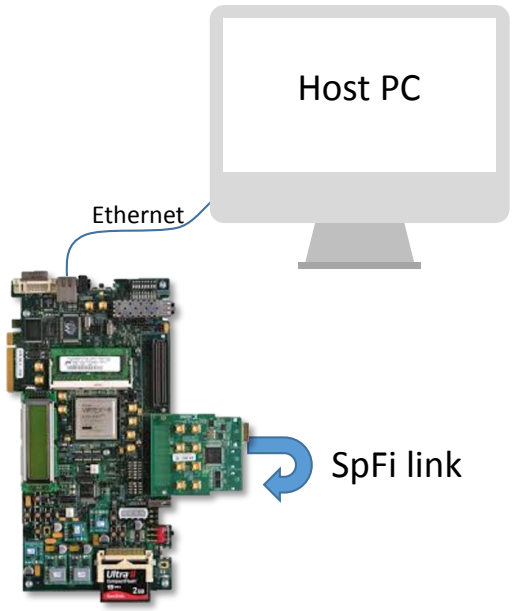


Fig. 4. SpaceFibre test equipment demonstrator connection for testing

Some of the most remarkable tests are:

i) Hardware packet generators and consumers were intensively tested with a lot of different packet sizes and bandwidths. Figure 5 shows the SpaceFibre test equipment GUI for the setup of the SpaceFibre port. The most useful parameters are directly visible on this window for all the Virtual Channels, so that the user can have a quick and complete overall view of the system. More specific parameters for each VC can be configured in a detailed section of the GUI.

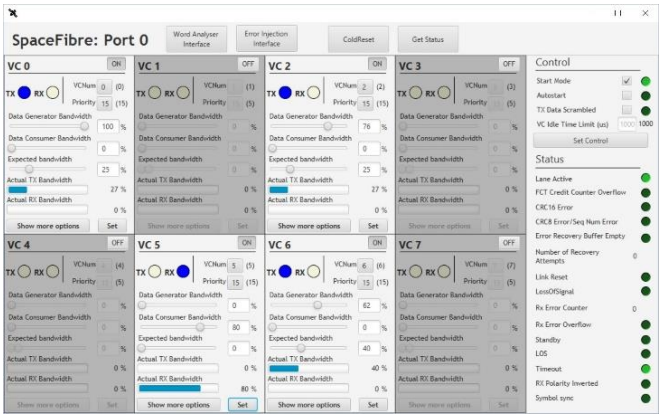


Fig. 5. GUI showing Hardware Packet Generator/Consumers on different Virtual Channels

ii) The Trace Memory was programmed to trigger in many different cases, both in transmission and in reception, completely exploiting and testing the trigger functionality of the component (Figure 6).

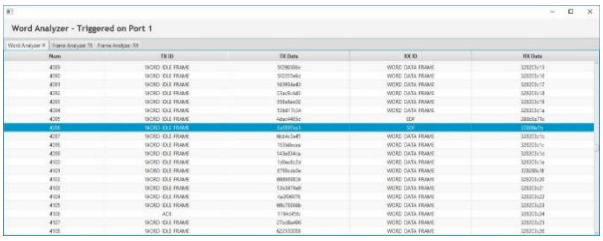


Fig. 6. Trace memory shown on the GUI

iii) The Error Injection module was greatly stressed. To verify its functioning the built-in trace memories of the SpaceFibre test equipment were used, testing all the three operation modes (bit flip, error on specific words and BER insertion on the link). The BER tests also verified the theoretical maximum error rate sustainable before the link disconnection (10^{-5}).

iv) The Software Packet Generator and Consumer were intensively used. This software tool generates a stream of fixed-step incremental data that is sent to a certain Virtual Channel of one of the SpaceFibre ports. By using the loopback connection, this stream is received by the other SpaceFibre port and sent back to the Host PC by the SpaceFibre test equipment unit. The software Packet Consumer also checks the correctness of the received data.

v) The software File Reader and File Writer were also used to test the SpaceFibre Analyser functionality. The working principle is the same of the Software Packet Generator and Consumer, but the data is read from a user-specified file and written back to another file. There is an option to read the file to send continuously, re-starting from its beginning when it ends, in order to stress more the test equipment. The correctness of the received data was proved by comparing the two files with external tools.

VI. Conclusions

A SpaceFibre test equipment, with its features and hardware architecture, was described in this paper. The proposed test equipment comes together with a complete Graphical User Interface that allows the user to extensively configure the unit itself and to put in place complex and automatically verifiable data transmission and reception schemes from and to the Host PC, using a variety of different interfaces to the host (Ethernet, PCIe).

The test equipment, with its two SpaceFibre and two SpaceWire ports, can be effectively used in many different situations, acting like an EGSE, or like a link analyser to monitor the link between two SpaceWire/SpaceFibre nodes, or performing some conformance tests to verify a third party SpaceWire/SpaceFibre device.

To accomplish these tasks, the SpaceFibre test equipment has many embedded tools:

- integrated Hardware Packet Generators and Consumers
- Trace Memory able to trigger on a user specified word and to show a window of the words flowing on the link
- Advanced Error Injection module able to insert errors in different modes (bit flips, word replacement, BER) and completely configurable by the user to artificially create particular operational scenarios.

Additionally, the test equipment is able to realize the bridge between the SpaceFibre and SpaceWire ports.

A great effort was spent in the configurability of all the aspects of the SpaceFibre test equipment by means of the GUI. The GUI, running on basically every OS, provides a compact yet complete view of every Virtual Channel of the SpaceFibre ports and provides to the user the complete control over all the aspects of the SpaceFibre test equipment. It also includes some very advanced functionality, like an automatic Software Packet Generator and Consumer/Checker and the possibility to send data to the Analyser reading from a file and, on the other hand, to write data received from the Analyser into a file.

All these features were deeply tested, making the SpaceFibre test equipment a powerful tool for the verification of external SpaceWire/SpaceFibre devices in many different situations.

REFERENCES

[1] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, "SpaceFibre Specification Draft H4", University of Dundee, April 2016

[2] D. Davalle, A. Leoni, L. Fanucci, "Implementation of a SpaceFibre CODEC compliant with the standard draft F3," 22nd SpaceWire Working Group Meeting, 2014

[3] D. Davalle, A. Leoni, L. Fanucci, "Verification environment for a SpaceFibre CODEC compliant with the standard draft F3," 23rd SpaceWire Working Group Meeting, 2015

[4] S. Parkes, C. McClements, D. McClaren, A. Monera, A. Ferrer, A. Gonzalez, "SpaceFibre Implementation, Test and Validation," 6th International SpaceWire Conference, 2014

[5] A. Ferrer, A. Gonzalez, C. McClements, S. Parkes, "STAR Fire: SpaceFibre diagnostic interface and analyser," 5th International SpaceWire Conference, 2013

[6] SpaceWire CODEC IP-core, IngeniArs S.r.l., http://www.ingeniars.com/english/products/space/spacewire-codec-ip-core.html

[7] Gigabit Serial Link Controller IP-core, IngeniArs S.r.l., http://www.ingeniars.com/english/products/space/gigabit-serial-link-controller-ip-core.html

A new Generation of SpaceFibre Test and Development Equipment

SpaceFibre, Short Paper

Alberto Gonzalez Villafranca, Steve Parkes,
Chris McClements, Bruce Yu, Pete Scott,
Albert Ferrer Florit

STAR-Dundee Ltd.
STAR House, 166 Nethergate, Dundee, DD1 4EE, UK
E-mail: alberto.gonzalez@star-dundee.com

***Abstract*— SpaceFibre is a new technology for use onboard spacecraft that provides point-to-point and networked interconnections at Gigabit rates with in-built Quality of Service and Fault Detection, Isolation and Recovery. The SpaceFibre standard is virtually finished, with the ECSS standardisation activity to be ended this year.**

There is a need for equipment to support the development and testing of applications of the entire protocol stack. This paper describes the new generation of SpaceFibre equipment designed for this purpose. They provide users with several options for platforms and connectors, such as FMC, USB 3.0, cPCI, PXI, PXIe and SpaceVPX. The number of platforms supported and the flexibility of the equipment provides the end user with a broad range of options to include SpaceFibre in their current system design. This helps to promote the adoption of SpaceFibre technology.

A number of designs using the equipment here described is currently available or under development. They include the SUNRISE SpaceFibre Router and the Multilane SpaceFibre interface, among others. When combined, these new boards and designs offer a powerful and rich set of tools to help with SpaceFibre designs.

***Index Terms*— SpaceFibre, RTG4, USB, cPCI, PXI, PXIe, SpaceVPX, EGSE, FMC**

I. Introduction

SpaceFibre (SpFi) will be released as an ECSS standard later this year [1]. With the addition of the network and multilaning layers the standard is virtually finished. STAR-Dundee released a few years ago the STAR Fire unit [2] to help with the SpFi implementation and adoption in the initial stages of the protocol. However, a new generation of Electronic Ground Support Equipment (EGSE) products is required to provide users with suitable hardware to implement and test the whole SpFi protocol stack. Furthermore, there is a need for demonstrators with space-qualified components to increase SpFi maturity. In this article a new family of products specifically designed to provide a platform to support SpFi adoption is presented.

II. STAR Fire Mk3

The STAR Fire Mk3 is the evolution of the initial STAR Fire device [2]. It shares with the old version some of its features. It has two SpFi and two SpaceWire (SpW) interfaces, two MICTOR connectors for connecting a Logic Analyser, and four SMB connectors. Three of those are external input triggers, and one is an external output trigger. Fig. 1 illustrates the block diagram of the STAR Fire Mk3 design. Fig. 2 shows the STAR Fire Mk3 unit.

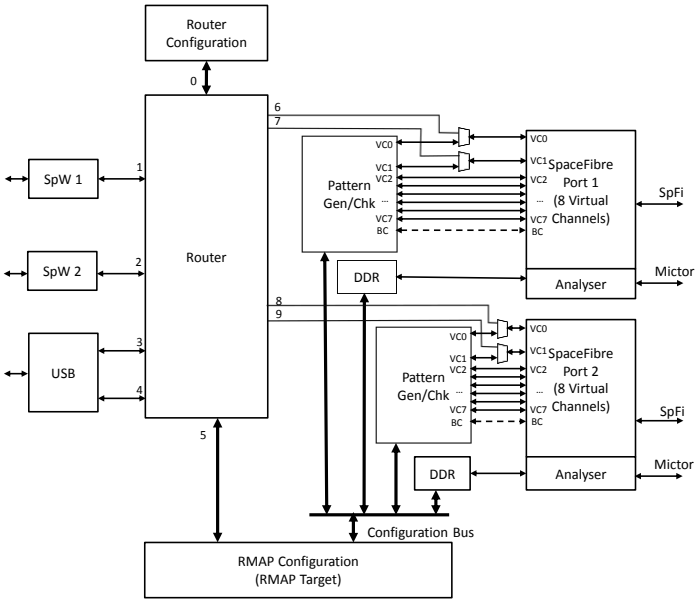


Figure 1. STAR Fire Mk3 architecture

The new STAR Fire unit can operate as a SpFi link analyser, SpFi interface and as a bridge between SpFi and SpW, among others. It has embedded pattern data generators and checkers. The Mk3 version features a USB 3.0 micro B interface, which provides communications with a much higher

data rate with the host PC. This means that the SpFi link can be directly interfaced from a computer at very high data rates. The old version can only use instead basic internal data generators and checkers for this purpose.

A bigger FPGA has also allowed an upgrade of the internal data generators and checkers to emulate realistic instruments. Specifically, this new unit features some of its Virtual Channels connected to advanced data generator and checkers. These provide with complex data generation capabilities, thus allowing more realistic data streams automatically generated and checked by the STAR Fire unit without the need for computer intervention. Some of the capabilities of these new data generators and checkers are:

- The type of data pattern can be selected among different options: random pattern, incrementing pattern, fixed word value, alternating word values, left or right circularly rotating four byte pattern
- Data value/seed is configurable
- Data pattern and packet lengths are configurable
- Length of data bursts and data rate can be configured
- EEP can be inserted in a specific position
- The initial four words of each packet can be configured

The capabilities of the embedded Analyser have also been improved. Now it is possible to trigger on any given data or control word received and also to select the Virtual Channel when triggering in data frames. Finally, a DDR memory is used instead of the internal FPGA memory to store the captured values, resulting in greater recording capabilities.

Figure 2. STAR Fire Mk3 unit

III. SpaceFibre PXI Board

PXI (PCI eXtensions for Instrumentation) is an industry standard widely used as a platform for electronic instrumentation in automated test systems [3]. It is currently used in many industry areas, including aerospace. PXI uses PCI in the communication backplane.

PXI Express (PXIe) uses the same PXI form factor but features PCI Express (PCIe) as backplane communication protocol. Switching from PCI to PCIe allows multiplying the available bandwidth from 132 MB/s up to 12 GB/s [4].

The SpaceFibre PXI board has been developed to implement a range of SpW and SpFi devices. The board is a 3U compatible with PXI, Compact PCI (cPCI) or PXIe racks. It can also be provided with the PXIe interconnection if required.

The board offers DDR memory and programmable clock sources to provide the end user with a very flexible architecture to implement multiple designs. It features a novel set of front panel interconnects. There is a set of flexible interface connectors that can be used to customise the board, such as SpFi, SpW, external triggers, etc. Thus, the board can be easily modified to accommodate different designs. This allows using the same PXI board to implement many different products.

Several designs have already been implemented using the PXI Board, such as the SUNRISE 8-port SpFi Router (Fig. 3), the STAR Fire design (Fig. 1), a 4-port SpFi interface, a Multilane (up to 4 lanes) SpFi interface, or a SpW to SpFi bridge.

Figure 3. *PXI Board configured as SpaceFibre Router. The front panel has 8 SpFi ports and 4 SpW ports*

A. The SUNRISE SpaceFibre Router

The SUNRISE router is the first implementation of a SpFi routing switch. Fig. 4 depicts the router block diagram. It features 8 SpFi interfaces with 4 Virtual Channels each, plus 4 SpW interfaces tied to a ninth SpFi port. All of them are accessible over the front panel, as shown in Fig 3. There is also an internal configuration port (Port 0).

This router implements path and logical addressing, group adaptive routing, virtual networks, time distribution and message broadcast. It also fully supports the Quality of Service (QoS) and Fault Detection Isolation and Recovery (FDIR) capabilities native to SpFi.

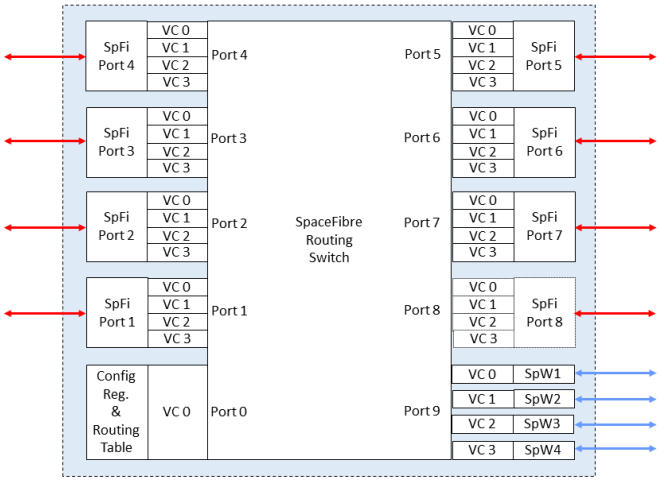


Figure 4. *SUNRISE SpaceFibre Router Block Diagram*

IV. SpaceFibre RTG4 PXIe Board

The SpaceFibre RTG4 PXIe board is a variation of the standard PXI board. This board is a 3U featuring a Microsemi RTG4 PROTO FPGA instead of a Spartan 6. This allows implementing designs with multiple SpFi and SpW interfaces in radiation-hardened technology.

Like the standard PXI, this board offers two banks of DDR memory, and a PXIe interface. It also offers the same set of flexible interfaces connectors as the PXI card. Up to 8 SpFi interfaces are supported. Furthermore, various front panel options are offered, also with an option for a custom front panel to support custom applications.

Current designs with this board include a multilane SpFi interface. Others designs planned for the near future include a 10-port SpFi router or an 8 lane-SpFi interface.

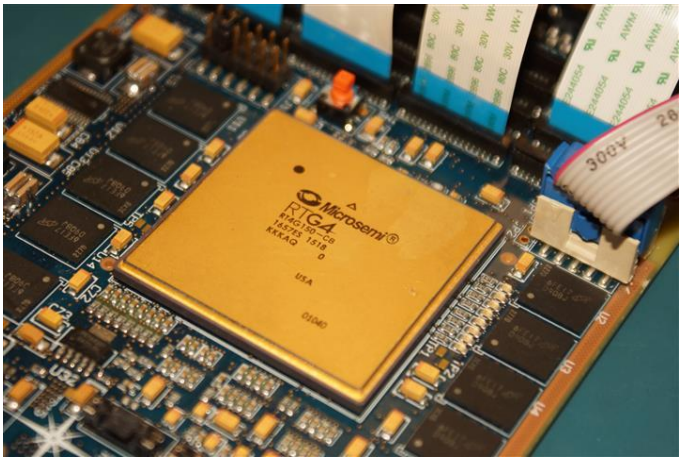


Figure 5. *SpaceFibre RTG4 PXIe board*

V. SpaceVPX-RTG4 Lite Board

SpaceVPX (also known as VITA 78) [6] uses the OpenVPX (VITA 65) backplane standard [5] adding features required for space to the VPX standard. They include important aspects in space, such as single-point failure tolerance, fault detection on critical configuration signals, robust system diagnostics, etc. Moreover, SpaceVPX offers the possibility of using SpW and SpFi for control and data planes.

A SpaceVPX Lite board will be made available in the coming months. Similar to the aforementioned PXIe-RTG4 board, this is a 3U unit with a Microsemi RTG4 PROTO silicon and a SpaceVPX [6] interface supporting a SpW control plane and a SpFi data plane along with standard management functions. In the front panel there are available two SpW and two SpFi connectors for user access.

An FMC daughterboard connector is available, with a family of daughter boards planned. Among them, a dual 3 GSamples/s ADC FMC board will be released with the board.

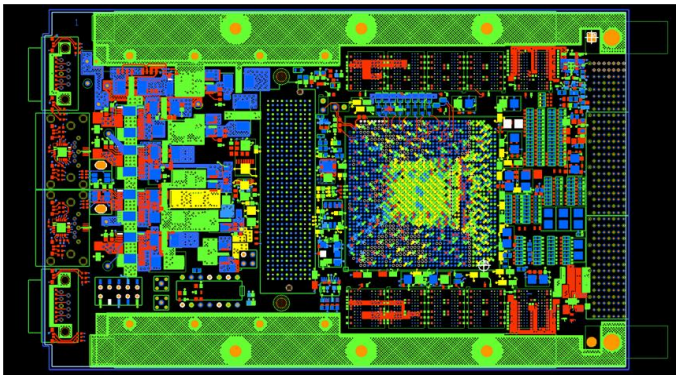


Figure 6. *SpaceVPX-RTG4 PCB design*

VI. FMC SpaceWire/SpaceFibre Board

The FMC-SpaceWire/SpaceFibre board (Fig. 7) is an FPGA Mezzanine Card (FMC) which is designed to extend the capabilities of an FPGA development board by adding support for SpW and SpFi interfaces. The board features a standard FMC High Pin Count (HPC) connector and has four SpW ports with accompanying tri-colour status LEDs, and two SpFi ports. The SpW signals are connected via LVDS buffers, and all SpFi signals are AC coupled. This adds protection preventing damage to the FPGA in case of signals levels being out of specifications.

There is an on-board 125 MHz oscillator that can be used as a reference clock inside the FPGA. Also, two SMA connectors provide with the option of using an external differential clock input instead of the on-board oscillator. 20 GPIO pins are available to the user.

Two sets of switches are used to set different connections of the SpW and SpFi signals on the FMC connector. The FMC board can be configured using the switches to work with a number of FPGA development kits including but not limited to:

- Microsemi RTG4 Development Kit - HPC1 and HPC2
- SmartFusion2 Adv Dev Kit - HPC and LPC
- Xilinx VC707/VC709 Board

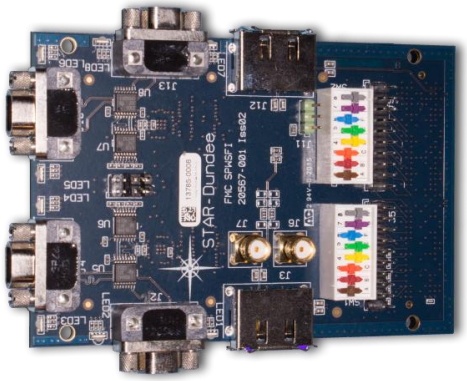


Figure 7. FMC-SpaceWire/SpaceFibre board

VII. Conclusion

The SpaceFibre standard is now basically complete. There is a growing number of space applications that can benefit from the SpFi features, namely, multi-Gbps data rate and in-built QoS and FDIR. Consequently, there is a growing interest on SpFi. A new set of products to support its adoption is required.

In this article a new generation of SpFi test and development equipment has been described. The equipment is flexible and supports popular platforms and connectors such as FMC, USB 3.0, cPCI, PXI, PXIe and SpaceVPX. Furthermore, a number of designs using this equipment is ready or under development and will be also made available by STAR-Dundee. These include the SUNRISE SpFi Router, Multilane or Multiport SpFi interfaces, etc. When combined, these new boards and designs offer a powerful and rich set of tools to help with SpFi designs.

References

[1] S. Parkes, A. Ferrer, A. Gonzalez and C. McClements, "SpaceFibre Standard", Draft H4, April 2016, available from https://indico.esa.int/indico/event/126/session/0/contribution/1 (last accessed 29th August 2016).

[2] A. Ferrer Florit, A. Gonzalez Villafranca, C. McClements, S. Parkes, "STAR Fire: SpaceFibre diagnostic interface and analyser", SpaceWire Conference 2013, Goteborg.

[3] PXI-1 Hardware Specification Revision 2.2, PXI Systems Alliance, September 22 2004.

[4] PXI-5 PXI Express Hardware Specification Revision 1.0, PXI Systems Alliance, August 22 2005.

[5] VPX: Base Specification, ANSI/VITA 46.0, www.vita.org, 2007.

[6] SpaceVPX Systems, ANSI/VITA 78.00, www.vita.org, 2015.

Networks & Protocols 1 (Long)

The Geostationary Operational Satellite R Series SpaceWire Based Data System

Session: SpaceWire Networks and Protocols, Long Paper

William Anderson
GOES-R Flight Data System Lead Engineer
NASA Goddard Space Flight Center
Greenbelt, MD USA
william.h.anderson@nasa.gov

Alexander Krimchansky
GOES-R Mission Systems Manager
NASA Goddard Space Flight Center
Greenbelt, MD USA
alexander.krimchansky@nasa.gov

Michael Birmingham
GOES-R Embedded Software Engineer
NASA Goddard Space Flight Center
Denver, CO USA
mike.j.birmingham@lmco.com

Matthew Lombardi
GOES-R Simulation and Test Engineer
Lockheed Martin
Denver, CO USA
matthew.s.lombardi@lmco.com

Abstract - The Geostationary Operational Environmental Satellite R-Series Program (GOES-R, S, T, and U) mission is a joint program between National Oceanic & Atmospheric Administration (NOAA) and National Aeronautics & Space Administration (NASA) Goddard Space Flight Center (GSFC). SpaceWire was selected as the science data bus as well as command and telemetry for the GOES instruments. GOES-R, S, T, and U spacecraft have a mission data loss requirement for all data transfers between the instruments and spacecraft requiring error detection and correction at the packet level. The GOES-R Reliable Data Delivery Protocol (GRDDP) [1] was developed in house to provide a means of reliably delivering data among various on board sources and sinks. The GRDDP was presented to and accepted by the European Cooperation for Space Standardization (ECSS) and is part of the ECSS Protocol Identification Standard [2].

GOES-R development and integration is complete and the observatory is scheduled for launch November 2016. Now that instrument to spacecraft integration is complete, GOES-R Project reviewed lessons learned to determine how the GRDDP could be revised to improve the integration process. Based on knowledge gained during the instrument to spacecraft integration process the following is presented to help potential GRDDP users improve their system designs and implementation.

I. INTRODUCTION

The GOES-R, S, T, and U spacecraft program is a key element of the National Oceanic and Atmospheric Administration's (NOAA) weather satellite observation operations. The GOES-R spacecraft uses European Cooperation for Space Standardization (ECSS) SpaceWire (SpW) [3] for the transfer of sensor, telemetry, ancillary, command, time code, and time synchronization information between instruments and the spacecraft. In addition, the spacecraft and instruments are required to use the GRDDP for all data transferred over on-board SpaceWire links.

In an effort to minimize risk the GRDDP underwent a robust testing program by the GOES-R Project, instrument providers, and spacecraft developer. Several SpaceWire router implementations were used in this testing as well final mission integration. These implementations included two different ASICs and three different FPGAs designs. Integrating this diverse combination of SpW routers proved to be a challenge for GOES-R. The current version of GRDDP will not be modified for the follow-on GOES spacecraft. Hardware and software for the GOES-S, T, and U spacecraft are copies of GOES-R. The construction and integration of these spacecraft have progressed to the point where it is cost prohibitive to make changes to their hardware and software.

II. CURRENT GRDDP FEATURES

The GRDDP uses the lower level SpaceWire data link layer to provide reliable packet delivery services to one or higher level host application processes. For the GOES-R series spacecraft, the lower level protocol is the Packet Level service specified in the ECSS SpaceWire standard [3]. The original GRDDP requirement goal was to have no data loss with a simple to implement protocol using microcontroller, ASIC, or FPGA designs.

The GRDDP design philosophy is that all good received packets must send an acknowledgement packet (ACK) to the transmitter. Packets with errors are discarded and not acknowledged. The header is eight bytes and has an eight bit CRC trailer. The protocol can be used in simple point-to-point full duplex interfaces or a full networked environment. GOES-R has both point-to-point and networked environments.

The GRDDP has two protocol services which are Reliable Delivery (RD) and Urgent Message (UM). The RD service requires a positive acknowledgement for all received error free packets. This service is used for data that is critical to the mission. Examples of RD data types are instrument sensor data, commands, and critical telemetry. UM service is for data that is fire-and-forget such as ancillary data and less critical telemetry that updates at higher rates. RD packets must utilize the header sequence number for sliding window, missing packet detection, duplicate packet detection, and packet order processing where UM packets do not.

There is a virtual channel capability included in the GRDDP. This allows up to 96 virtual channels (VC) to be used on a single physical SpW link. VCs are defined by SpW Logical Address (SLA) pairs that are required to operate independently. The VC capability allows mixing of low, medium, and high rate data on a single physical SpW connection, while having a logical separation.

Packet segmentation is not allowed in order to comply with the keep-it-simple philosophy for the protocol. If a user has shorter packets it is allowed to pack out the 64k application space. Any error condition outside the protocol's ability to manage it causes the GRDDP to stop and report this condition to a higher level process.

III. GRDDP STARTUP REQUIREMENTS MISSING FROM ORIGINAL SPECIFICATION

There are five instruments on the GOES-R spacecraft covering a wide range of Earth and Solar sensing capabilities. The data rates from these instruments varies widely. The instruments started development before a spacecraft contractor was selected. This created a situation where the spacecraft provider wasn't in a position to negotiate instrument operation over the SpW GRDDP links. The instrument providers all implemented the startup operation of their GRDDP interfaces differently since the specification was not explicitly specific. This caused the spacecraft to develop unique startup processes for each instrument.

During GRDDP development and proof-of-concept testing, the GOES-R Project development system host processors were up and running long before instrument links were established. This case was reversed when the instruments were integrated with the spacecraft. The instrument's GRDDP interfaces established links with the spacecraft before their host processors were fully functional. Another difference between GRDDP development and spacecraft integration is that development system startup procedures were implemented manually where the spacecraft procedures are automated. This prevented testing during development that could have identified startup interface failures due to host processor initialization delays. Additional problems relating to instrument reset/reboot and timecode processing were encountered.

Post GRDDP development and when the instrument providers delivered emulators, integration and software verification testing proceeded using spacecraft host processors and simulators. The spacecraft simulators had functionally equivalent spacecraft SpW network routers and were used to test command products and procedures in operational configurations. This testing revealed problems with instrument reset/reboot, router flow control, and timecode processing. These issues were analyzed and resolved.

The primary and highest rate GOES-R instrument GRDDP interface establishes links with the spacecraft and buffers messages until the instrument's host processor is ready. It takes approximately 3 seconds for this instrument's host processor to reach the state where it can configure the GRDDP interface and process messages. Due to GOES-R timecode processing requirements, the instrument needs to receive time information from the spacecraft as soon as the link is established. In order to deal with these issues procedures were created holding

high rate and ancillary packets for 3 seconds. No delays were implemented for time messages.

The link to host processor startup latency issue was a problem for all GOES-R instruments. The corrective action implemented, shown in Figure 1, was creation of a programmable delay in the instrument startup procedures holding off link initialization until the host was fully operational. This programmable delay is different with each instrument and defined in spacecraft to instrument Interface Control Documents (ICD).

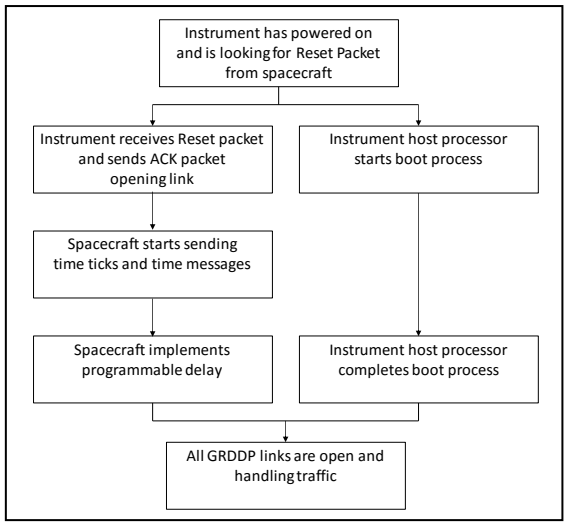


Figure 1. GOES-R GRDDP Startup Host Delay Compensation Procedure

The next highest rate GOES-R instrument is the first of four instrument nodes, as seen from the spacecraft, in an onboard network. This instrument designed their flight software to initiate an internal reset every 1.3 seconds if a spacecraft transmit channel is not opened causing a disconnect/reconnect cycle. This causes the SpW timecode hardware to lose synchronization with the other nodes on the router, since a disconnect on any port resets the six bit SpW timecode count for all ports. If this instrument's internal resets continue, the other instruments on the link will see a loss of timecode for a period of greater than ten seconds. This condition, by GOES-R requirements, causes the other instruments on the link to safe themselves. To mitigate this problem, procedures were developed that powered this instrument on and opened GRDDP channels before the other instruments sharing the same link were powered on and activated. Also, procedures diagramed in Figure 2 were implemented that powered this instrument down after shutting down the other instruments on the link.

Based on GOES-R integration experience as discussed above several modifications to the GRDDP specification are recommended. In order to deal with link to host initialization delays GRDDP users should revise the protocol with the following:

From:

> **7.2 Reset Command**
> When a Transmit End Point (TEP) transitions to the Enabled state, it **shall** send a Reset command to its remote Receive TEP and initiate an acknowledgement timer.

To:

> **7.2 Reset Command**
> When a Transmit TEP transitions to the Enabled state *and all associated processors are fully operational*, it **shall** send a Reset command to its remote Receive TEP and initiate an acknowledgement timer.

This compensates for link to host startup delay issues and eliminates the need for programmable delays in startup procedures.

GOES-R has implemented a dual redundant SpW architecture between the spacecraft and onboard instruments. If GRDDP users implement redundant SpW interfaces GRDDP requirements need to be added to insure transmitters and receivers are connected to the correct link.

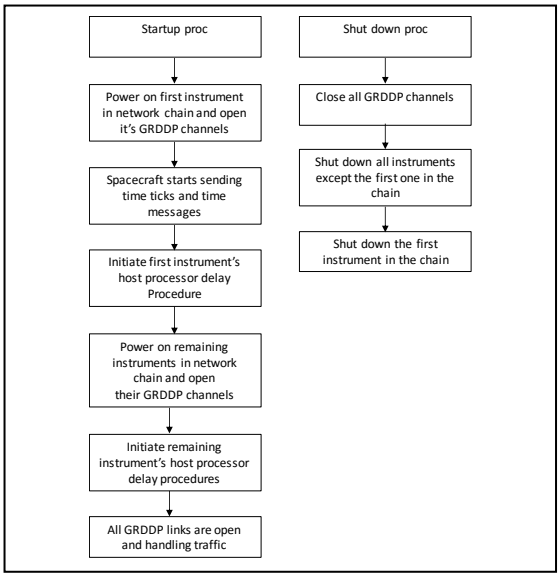


Figure 2. Startup and Shutdown Procedure

IV. GRDDP PRIORITY PROCESSING

The current GRDDP requirements for transmit priority processing are as follows:

4.1.2 Transmit Priority

When more than one packet is available for transmit, all Acknowledge packets shall be transmitted first, then Reset Command packets, then Urgent Message packets, then retransmit packets, then Data packets.

4.1.3 Data Transmit Queue

When data packets from more than one channel are available for transmit, packets shall be transmitted in the order in which they are queued.

4.1.4 Urgent Message Transmit Queue

When Urgent Message packets from more than one channel are available for transmit, packets **shall** be transmitted in the order in which they are queued.

The GOES-R spacecraft uses the GRDDP UM service to distribute ancillary packets at a 100Hz rate to the primary instrument. There is a latency requirement for this ancillary data. The spacecraft implemented flight software compliant with the GRDDP specification. Due to a unique set of circumstances, this instrument's operational procedures caused delays in transmitting ancillary data packets outside the latency requirement. It was determined the ancillary packet latency requirement was more important than meeting GRDDP priority requirements. A modification was made to transmit packets that met the latency requirement and violated GRDDP requirements.

It is recommended that GRDDP transmit packet priority processing requirements be changed to allow a more adaptive design. The basis for this requirement change is that when a packet is ready and the channel is idle it should be sent immediately instead of being sent to a queue. In the case where the channel is busy the packet should be sent to a queue. As soon as the channel returns to an idle state queued packets need to be transmitted highest priority first.

V. HEADER REVISION

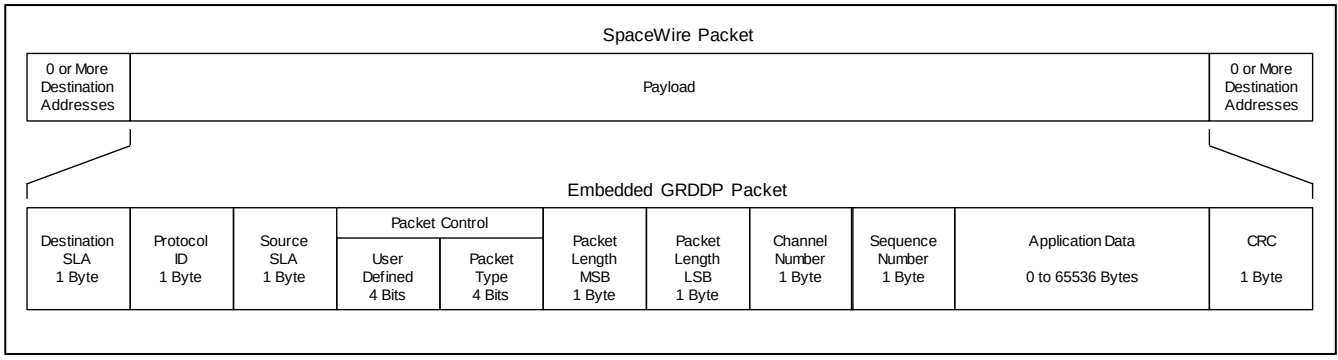
The current GRDDP specification defines a single header format for all protocol packet types and is shown in Figure 3. Based on experience gained from GOES-R spacecraft and instrument integration a revision to the data packet header is proposed. In addition, an ACK packet and Reset packet header are to be added to the GRDDP. These changes enable the protocol to be more robust. Also, these recommendations improve error detection and management capabilities.

The first of these recommended header revisions is addition of a version number to all three packet types. The version number aids in detecting packets that may be in the data stream, but are not valid for a specific mission need. Inclusion of a version number replaces the user defined nibble in the original GRDDP header. The proposed GRDDP Data packet header is shown in Figure 4.

Originally it was intended that the spacecraft and instruments cooperatively develop and Interface Control Document (ICD) defining GRDDP user selectable parameters. It was assumed the spacecraft to instrument ICD would be adequate and allow "trouble free" spacecraft to instrument communications. However, as spacecraft and instrument development advanced into integration mismatches occurred. The reason for these mismatches was due to the lack of a requirement for either side to verify the other side's operation. The Reset packet header format is to be lengthened by 5 bytes and is shown in Figure 5.

This proposed Reset packet header allows a plug-and-play environment where programmable protocol parameters would be provided to protocol receivers. The receiver could optionally adaptively configure for that channel's parameters. The ACK to the Reset packet could either verify matching parameters to a predetermined configuration or indicate some requirement is beyond the receiver's capabilities.

Figure 3. Current GRDDP Header



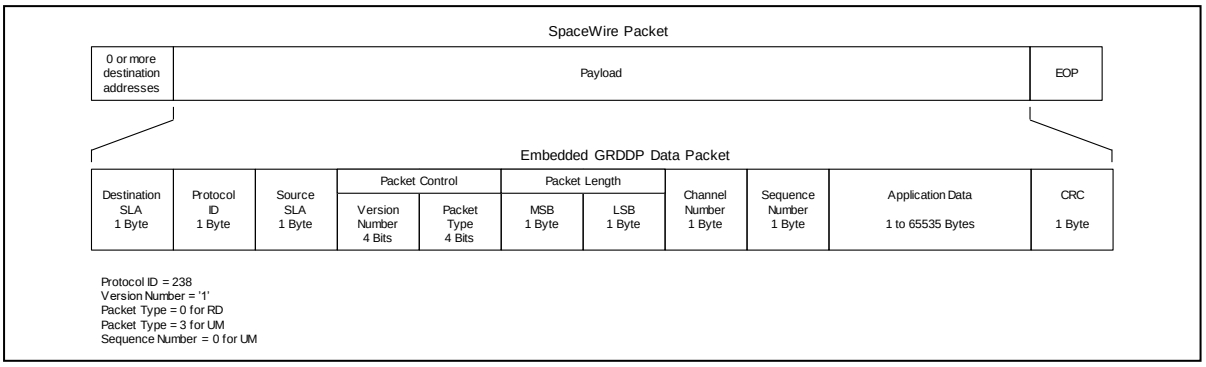


Figure 4. Proposed Data Packet Header

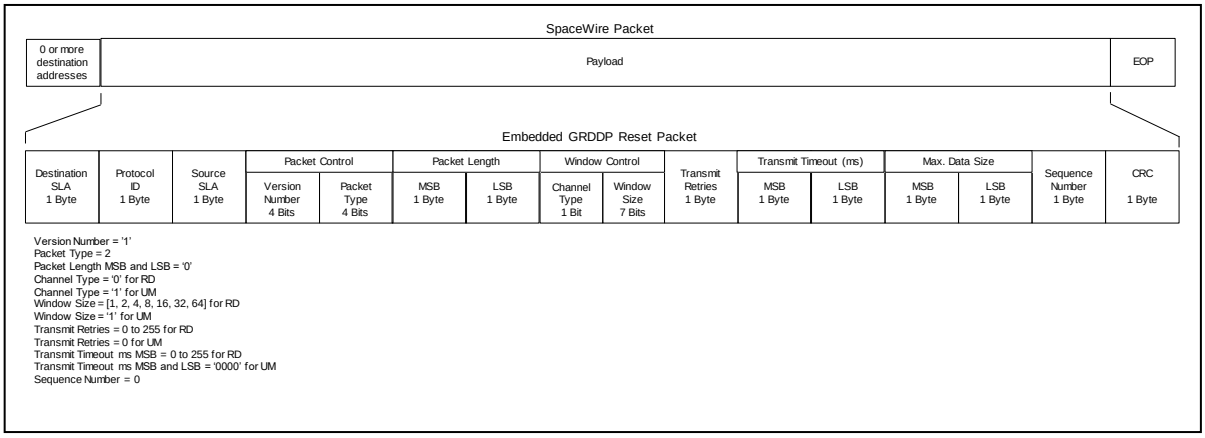


Figure 5. Proposed Reset Packet Header

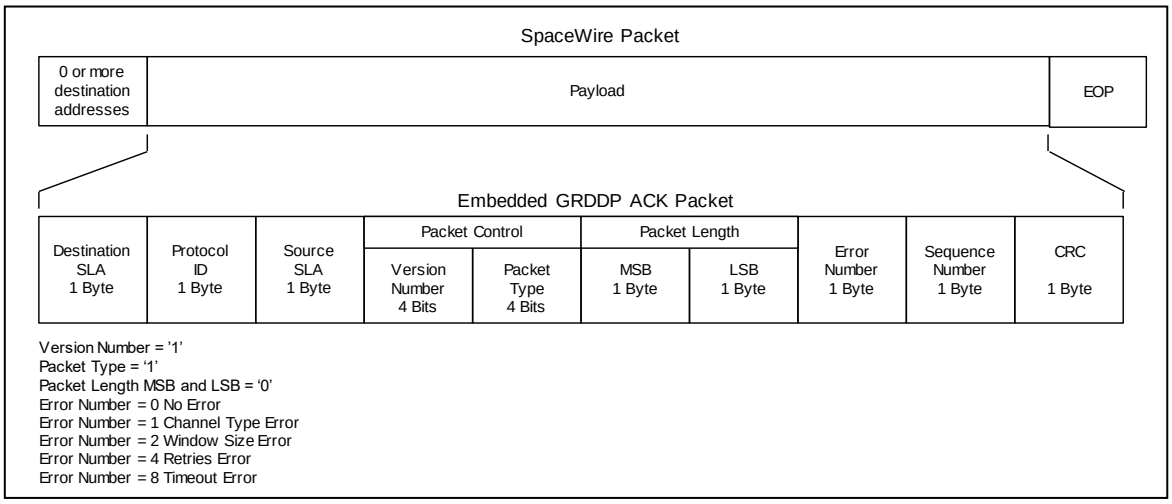


Figure 6. Proposed ACK Packet Header

VI. MISCELLANEOUS RECOMMENDATIONS

There is no current GRDDP requirement that a channel should exclusively deliver RD or UM packets. In order to reduce the possibility of complications, it is recommended that such a requirement be added. Reset packet rate has proven to be a problem. It is recommended that a requirement be added controlling GRDDP Reset packet rate to something on the order of once a second.

It is essential that the spacecraft disable the redundant port to the instrument prior to power on. Also the instrument should initiate communications after determining which side is active based on the link's run status. In order to eliminate programmed delays in power up sequences the spacecraft needs to detect when the instrument is "alive." This is accomplished when the spacecraft receives a Reset packet. The next step is to open timecode and command channels. When all channels are in the Open state telemetry data should commence.

REFERENCES

[1] NASA Goddard Space Flight Center GOES-R Project "GOES-R Reliable Data Delivery Protocol," 417-R-RPT - 0050 Version 2.1, 2008

[2] European Cooperation for Space Standardization, ECSS-E-ST-50-51C 5, "Space Engineering SpaceWire Protocol Identification," February 2010

[3] European Cooperation for Space Standardization, ECSS-E-50-12A, "SpaceWire – Links, Nodes, Routers and Networks," 2003

Streaming Services over SpaceFibre Networks

SpaceWire networks and protocols, Long Paper

Ilya Korobkov, Elena Suvorova, Yuriy Sheynin, Valentin Olenev
Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
{ilya.korobkov, valentin.olenev}@guap.ru, {suvorova, sheynin}@aanet.ru

***Abstract* — Modern and prospective spacecraft data system networks consist of many systems and sensors producing streaming traffic. Outside spacecraft video cameras also generate intensive data streams. Motion video traffic requires specific latency and speed. Video frames should be delivered with small delays and jitter over high-rate SpaceFibre networks.**

The paper considers live streaming video over onboard spacecraft networks with its fixed packet size and periodical issue, detection of packet reordering, small delays. ARINC-818-2 and CCSDS Digital Motion Imagery streaming traffic, characteristic of video streams are analysed, requirements for streaming services and transport protocol are presented, the overview of existing streaming protocols is done.

The STP-2 protocol was proposed for streaming data service in SpaceFibre networks. It is based on STP, which provides a number of native streaming features. STP-2 has some significant modifications that improve delivery of streaming data flows over high-rate SpaceFibre networks. Use cases for it application illustrate its benefits.

***Index Terms* — Spacecraft, Onboard Networks, Live Video, Streaming Features, SpaceFibre, STP-2.**

I. INTRODUCTION

Modern communications and data exchange between onboard spacecraft systems could be cyclic [1] and streaming [2, 3]. This is especially true for optical and television systems, for example, onboard video camera. Live streaming video forms special requirements for data delivery speed, latency and jitter.

To provide necessary bandwidth for transmitting video streams over onboard spacecraft networks high-speed SpaceFibre networks may be used [4]. SpaceFibre has a compatibility with SpaceWire standard on Network level. SpaceFibre specification describes only three bottom layers of the OSI model and does not cover transport layer and streaming service. Therefore, there is an essential task to analyze existing transport layer protocols for streaming data, taking into account characteristics of video streaming traffic and the compact implementation in spacecraft onboard systems.

To solve this problem the research of streaming traffic and its features was done. The industrial aerospace standards for streaming video ARINC-818-2 [5] and CCSDS Digital Motion Imagery [6], characteristic of video streams are analysed, requirements for streaming services and transport protocol are represented, overview of existing streaming protocols is done.

II. STREAMING TRAFFIC: DEFINITION AND MAIN FEATURES

There are several definitions of streaming traffic:

1) Traffic type characterized by viewing and/or listening for information as new information becomes available [7].

2) Streaming data transfer is a way of transferring real-time or buffered data such as sound, video, documents or photos through the networks with acceptable Quality of Service. Receiving system can start playback or display data before receiving full information [8].

3) Streaming traffic is the uniform data stream with a constant bit rate [9].

The third definition is more general but, it is the most representative for streaming traffic. There are several main features of streaming traffic: fixed packet size (no wide range of sizes); periodic packet issue with stable intensity; tolerant to single and sporadic corruption; allow to predict buffer size, optimise and streamline sender\receiver equipment [2].

Here is a short summary of industrial standards for streaming video ARINC-818-2 and CCSDS 766.1-B-1 Digital Motion Imagery.

III. SUMMARY OF ARINC-818-2 AND CCSDS 766.1-B-1

ARINC-818-2 (ARINC) is the standard that describes interfaces for transmitting video information to cockpit displays of civil and military aircrafts: Boeing 787, Airbus A380, A400, C-130, F18, F22, F35. ARINC specification describes video/audio transmission in real-time, information indicating on the pilot cabins displays [5]. Transmitted data can be uncompressed, compressed or encrypted. ARINC standard includes information about parameters of transmitting video (TABLE I.).

TABLE I. PARAMETERS OF VIDEO TRAFFIC IN ARINC-818-2

Resolution		***Frame size, Kbyte**	***Line size, Kbyte**	**Playback frequency**	**Accepted latency**
VGA	640x480	600	1,25	15 - 120 Hz	CCSDS 2015 conference: 50 ms – telerobotics
SVGA	800x600	937,5	1,56		
XGA	1024x768	1536	2,00		

Resolution		*Frame size, Kbyte	*Line size, Kbyte	Playback frequency	Accepted latency
WXGA	1366x768	2049	2,67		NASA; 60, 75 ms – critical communication; 100 ms – interactive video.
SXGA	1280x1024	2560	2,50		
SXGA+	1400x1050	2871,1	2,73		
WSXGA	1600x1024	3200	3,13		
UXGA	1600x1200	3750	3,13		
1920x1440		5400	3,75		

* - results are for 16 bit color depth.

CCSDS 766.1-B-1 Digital Motion Imagery (hereinafter referred to as CCSDS) is a standard that identifies which television and video industry standards should be utilized for interoperability in a spacecraft, between spacecrafts and between a spacecraft and Earth. The CCSDS specification describes real-time video data transmission and video streaming (telecasting). Transmitted data can be uncompressed, compressed or encrypted (Secure JPEG2000) [6]. CCSDS includes information about parameters of transmitting video (TABLE II.).

TABLE II. PARAMETERS OF VIDEO TRAFFIC IN CCSDS 766.1-B-1

Traffic	Resolution	*Frame size, Kbyte	*Line size, Kbyte	Playback frequency, Hz
Personal video conferencing	320x240..1280x720	150..1800	0,625..2,5	10 – 60
Medical conferencing	320x240..1280x720 Standard resolution 640x480	150..1800 600	0,625..2,5 1,25	10 – 60
Situational awareness	640x480..1280x720	600..1800	1,25..2,5	25 – 60
Public affairs				24, 25, 60
High Resolution Digital Imaging	1920x1080..4096x2160	4050..17280	3,75..8	24 – 120

* - results are for 16 bit color depth.

IV. REQUIREMENTS FOR ONBOARD STREAMING DATA DELIVERY PROTOCOL

Basing on the analysis of definitions and features of the streaming traffic, industrial standards ARINC-818-2 and CCSDS 766.1-B-1, streaming video characteristics and requirements of aerospace industry to compactness and simplicity of onboard systems implementation, following requirements for the onboard streaming protocol were formulated:

- Stable intensity of packet issue: It is supported by fixed packet size (excluding some cases, such as compressed video) during the communication session and fixed period of packet issue. It follows from the streaming definition [9];
- Small delays of the real-time streaming data transmission:
  - *a)* a connection oriented protocol allows to reduce the header size for data packets with the payload;
  - *b)* simple data delivery mechanism implemented at the hardware level:
    - No buffering on the sender and receiver sides;
    - No acknowledgements and retries;
- Data delivery control on the receiver side:
  - *a)* Check packet header for correctness;
  - *b)* Packet filtering – dropping the packets with error header;
  - *c)* Packet loss and out-of-order detection;
- Compatibility with SpaceFibre/SpaceWire.

V. STREAMING ORIENTED PROTOCOLS OVERVIEW

Detailed overview of existing streaming protocols was done. It is based on researches [2, 10, 11, 12, 13]. Following protocols were considered:

- Internet, multimedia and real-time Transport layer protocols: TCP, UDP, RTP, RTCP, SCTP, SSTP, RSVP, DCCP;
- Onboard and aerospace Transport layer protocols: Saratoga, ECSS-E-50, CFDP, SCPS-TP, JRDDP, STP, STP-ISS rev.2;
- Protocol stacks that can be used for streaming: SOIS, RapidIO, ARINC-818-2, SpaceWire, SpaceFibre.

Also streaming protocols of Application layer (such as Apple HLS, Adobe RTMP and others) were reviewed as general purpose mainstream streaming protocols. Short description for each considered protocols are given in [14].

Streaming features of each reviewed onboard and aerospace transport protocols are presented in TABLE III. – TABLE VI.

TABLE III. COMPARISON OF INTERNET PROTOCOLS AND REAL-TIME TRANSPORT LAYER PROTOCOLS

Mechanisms and features	TCP	UDP	RTP	RTCP	SCTP	SSTP	RSVP	DCCP
Header length, bytes	20-60	8	16	Transmits data transfer reports	12	16	8	12-16
Max payload, bytes	64K	64K	Depends on the profile		64K	1G	64K	1020
Compatibility	IP networks		UDP			IP networks		
Fixed packet size	-	-	-	+	-	-	-	-
Periodical data transfer	-	-	-	+	-	-	-	-
Best-effort delivery	-	+	+	+	-	+	+	+
Data correctness check	+	-	+	-	+	+	-	+
Data sequence check	+	-	+	-	+	+	-	-

Mechanisms and features	TCP	UDP	RTP	RTCP	SCTP	SSTP	RSVP	DCCP
Only without packet retransmission	-	+	-	+	-	-	+	+
Time stamp in packet	-	-	+	+	+-	-	-	-

TABLE IV. COMPARISON OF ONBOARD AEROSPACE TRANSPORT LAYER PROTOCOLS

Mechanisms and features	SARATOGA	ECSS-E-50-13	CFDP	SCPS-TP	JRDDP	STP	STP-ISS rev.2
Header length, bytes	12	2	4	15	10	8	9
Max payload, bytes	$256 \cdot 10^9$	4K	64K	64K	64K	4G	2K or 64K
Compatibility	UDP/ UDP-lite	MILSTD-1553B	SCPS-SP, IPSec, IPv4/v6	SpaceWire or SpaceFibre			
Fixed packet size	+	-	+	-	-	+	-
Periodical data transfer	-	-	-	-	-	+	-
Best-effort delivery	+	+	+	+	+	+	+
Data correctness check	+	+	+	+	+	+	+
Data sequence check	+	-	+	+	+	-	-
Only without packet retransmission	-	-	-	-	-	+	+-
Time stamp in packet	+	-	-	-	-	-	-

TABLE V. COMPARISON OF INTERNET AND MULTIMEDIA APPLICATION LAYER PROTOCOLS USED FOR STREAMING

Mechanisms and features	RTSP	Adobe RTMP	MPEG-TS	Apple HLS	Adobe HDS	Microsoft SS
Header length, bytes	Transmits control commands to video server	18	4	4 (MPEG-TS)	$16 \cdot 10^9$ (max. size MP4)	
Max payload, bytes		16M	184	184 (MPEG-TS)		
Compatibility	RTP, UDP, TCP	TCP	any transport networks	HTTP		
Fixed packet size	-	+	-	+	+	+
Periodical data transfer	-	-	-	-	-	-

Mechanisms and features	RTSP	Adobe RTMP	MPEG-TS	Apple HLS	Adobe HDS	Microsoft SS
Best-effort delivery	-	+	+	+	+	+
Data correctness check	-	-	-	-	-	-
Data sequence check	-	+	+	+	+	+
Only without packet retransmission	+	+	+	+	+	+
Time stamp in packet	-	+	+	+	+	+

TABLE VI. COMPARISON OF HIGH-PERFORMANCE PROTOCOL STACKS

Mechanisms and features	Spacecraft Onboard Interface Services	RapidIO	ARINC-818 (rev. 2)	SpaceWire	SpaceFibre
Protocol Data Unit (PDU)	Depends on the transport and data link layer protocols	Data Streaming packet	Fiber Channel frame	Packet	
PDU header length, bytes		4-8	24-28	Unlimited	
Max PDU payload, bytes		64K	2112		
Function	Spacecraft	Avionics		Spacecraft	
PDU fixed size	+-	-	-	-	-
Periodical data transfer	+-	-	-	-	-
Best-effort delivery	+	+	+	+	-
Data correctness check	+	+	+	+	+
Data sequence check	Depends on the transport and data link layer protocols	+	-	-	+
Only without packet retransmission		-	+	+	-
Time stamp in PDU		+	-	-	+

According to the conducted analysis, nowadays there is no streaming protocol which provides all required mechanisms for onboard streaming data delivery. Existing protocols were designed for specific tasks in general purpose applications.

Only the STP protocol [15] most closely meets onboard streaming protocol requirements: periodical packet issue and fixed packet size; packets are delivered without acknowledgments (best-effort delivery) and without lost packet retransmission; data correctness check is provided; it is connection oriented protocol; it has compatibility with SpaceFibre/SpaceWire. Thereby it was decided to modify existing STP protocol for solving essential problem of

streaming data delivery in onboard spacecraft SpaceFibre networks. Modified protocol was called STP-2. Description of main mechanisms of this protocol is provided in this paper.

VI. STP-2 – Streaming Transport Protocol edition2

A. General description

The STP-2 protocol is a transport layer protocol over SpaceFibre. It provides transmission of streaming data between nodes of a SpaceFibre network with stable intensity and fixed packet size (or fixed maximal packet size for application such as transmission of compressed video).

Data packets are used for streaming data delivery in STP-2. Data packets are delivered without acknowledgments and retransmissions. It may have a fixed size or a variable size. This allows using STP-2 for streaming traffic transmition with PDUs of the same size (for example, data from sensors, video frames, or lines of uncompressed video) or PDUs of the variable size (compressed video).

STP-2 protocol is a connection-oriented protocol. It supports up to 4096 transport connections for one device. Connection parameters are set in the transport connection establishment phase. Therefore STP-2 provides an ability to transmit large sized data with minimum overheads. Transport connection is established under the control of a master node. There are two main operating schemes:

1. Data exchange between two devices, one of which is the master and the second – slave. It supports data transfer from slave to master and from master to slave (Fig. 2. .

2. Data exchange directly between two slave devices (transmitter and receiver) under control of the third device – the master node (Fig. 3. .

STP-2 was developed for compact implementation in spacecraft onboard systems. There is no full packet buffering at the receiver and transmitter side. Implementation of STP-2 may be either completely hardware or hardware/software. To improve performance it is recommended to implement data send/receive STP-2 mechanisms in hardware; the transport connections control could be done in software.

STP-2 protocol has failure detection and indication mechanisms. The protocol detects errors in header and in payload of packets; packet loss (for data packets and servise command/packets). It is also monitored the failure of a master/slave device (as a result of the device crash or the communication link disconnection between them).

B. STP-2 Interfaces

There are two STP-2 interfaces with the Application layer: the streaming data interface and the configuration interface (Fig. 1.). The Streaming interface is used to transfer streaming data from applications. It is recommended to implement this interface in hardware for high performance. The Configuration interface provides means for the STP-2 configuration parameters change, for transmission of status information, reset commands.

The SpaceFibre packet interface is used for transmission of STP-2 packets over SpaceFibre virtual channels.

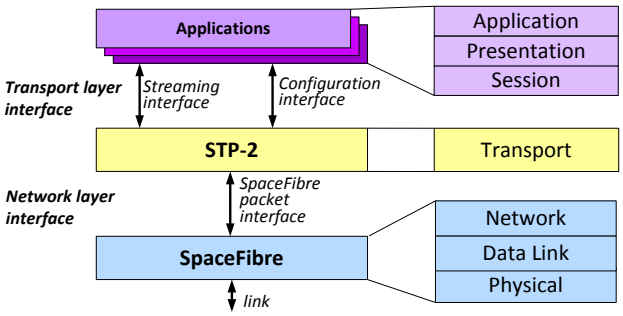


Fig. 1. The STP-2 protocol and OSI model

C. Basic principles of data exchange in STP-2

The Exchange data should be carried out in three phases:

1. Connection establishment.
2. Data transmission.
3. Connection closure.

Connection establishment and connection closure are performed under the master control in both STP-2 operating schemes.

There are two packet types in STP-2 protocol: service packets and data packets. Data packets are used to deliver streaming data. A Data packet can transfer up to 32M bytes. Service packets are used to establish and close connection, for flow control. There are several kinds of service packets:

- Transport connection control packets (Open Connection Request, Open Connection Confirm, Close Connection Request, ets.);
- Heart Beat – notify that a node is valid;
- Status Request – request to get statistic report for STP-2 transport connections;
- Status – transport connection statistics report: number of received/discarded packets, etc.
- Start – request to start data transfer from transmitter;
- Stop – request to stop data transfer from transmitter.

Receiver can control of data flow with using of Start and Stop commands.

The data exchange between a master and a slave is shown in Fig. 2. The master has initiated the connection establishment. The master is the data receiver in this example. The slave is data transmitter.

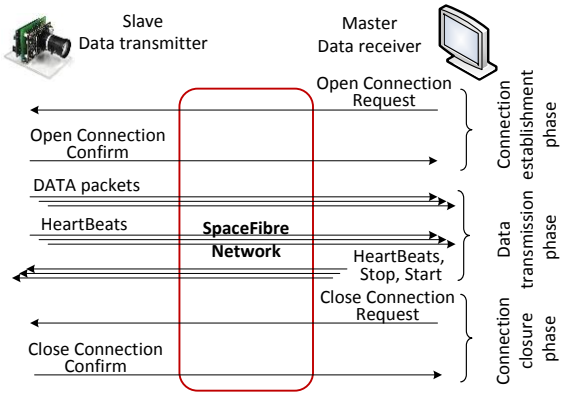


Fig. 2. The Data exchange between master and slave

The second STP-2 operating scheme is presented in Fig. 3. First of all the master configures transport connection for the slave receiver via the Open Connection Request (OCR) packet. Then the slave transmitter is configured by the master (see Fig. 3.).

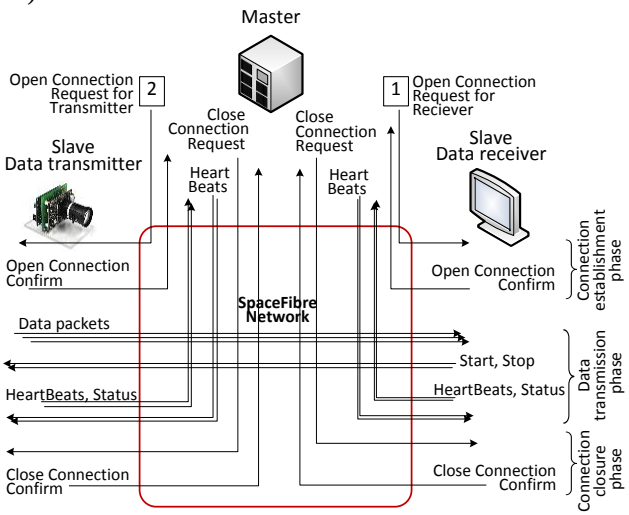


Fig. 3. The Data exchange between two slaves and remote master

D. Transport Connection Establishment and Flow control

STP-2 provides connection-oriented data transmission. There is a master device on one side of the connection and a slave device – on the other. Only master can be an initiator of the connection establishment. The transport connection establishment is performed by means of two-phase handshake (see Fig. 2). Initiator sends the OCR packet. This packet includes transmission parameters for the slave: connection ID (to identify connection in the master device and in the slave device), period of packet issue, max payload size, data transfer direction, etc. (Fig. 4.).

The slave responds with confirmation – Open Connection Confirm (OCC) packet. Then the t_{cnf} timer should be started. Timer expiration means that data transmission from the slave should be started.

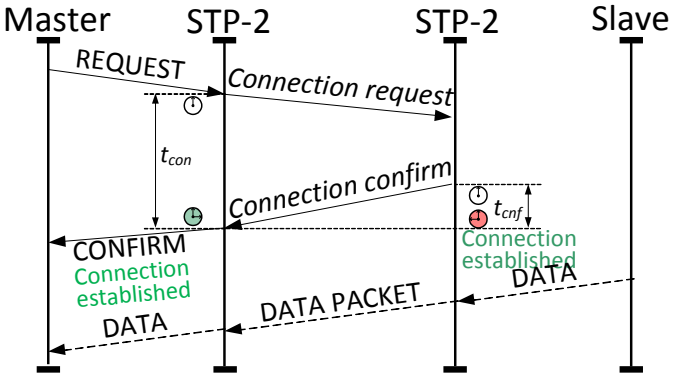


Fig. 4. Two-way handshake connection establishment phase in STP-2

The SpacerFibre protocol supports guaranteed data delivery at the Data link layer. But the OCC packet could be not deliver due a physical connection lose, routers failure or a failure in the slave device. For detection of this problem a connection timer (t_{con}) is used in the master. It startes when the OCR packet is sent.

If during t_{con} timer the OCC packet from the remote device has not been received, then the master device should resend the OCR packet. (The master can perform muliple retries of a connection establishment, since data transmission via the network can be restored, for example, due using of spare connection lanes and routers.) The slave could receive the repeated OCR packet from same master (in case the OCC packet from this slave is lost in the network). In such case the slave should recend the OCC packet.STP-2 should write information about this error to status register.

Flow control.

The OCR packet has the «Ready to receive» field. This field indicates that the master is ready to receive data from the remote device as soon as possible.

Then the transmitter sends packet after packet in the preset periods without waiting for any credits or confirmation from the receiver.

To limit streaming flows in the Data transmission phase STP-2 implements Stop and Start service packets. The Stop packet will stop data transmission (without connection closure). The Start packet will start (restart) data transfer from the transmitter.

E. Transport Connection Closure

Transport connection could be closed on request only from the master (Fig. 5.). The Master sends Close Connection Request (CCR) packet. Slave responds it – Close Connection Confirm (CCC) packet. t_{cls} and t_{end} timers are used in similar way like t_{con} and t_{cnf} timers during connection establishment phase.

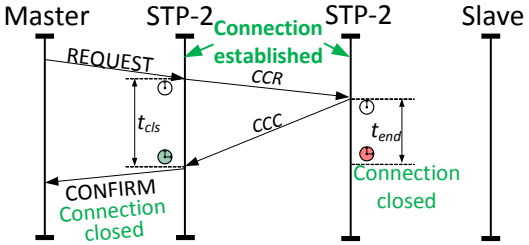


Fig. 5. The transport connection closure on request from master device

The CCR or the CCC packet can be lost in the network (as the OCR or the OCC can be lost in the Connection closure phase). But the CCR packet should not be repeated in this case. When the master goes to the Connection Closure phase, it stops transmission of the HeartBeat packets. If the CCR packet will be lost in the network, non-availability of HeartBeats would cause connection closure by the slave. If the CCC packet is lost in the network, the slave received CCR command and close connection.

Connection should be also closed by slave when there are no data packets or Heart Beat packets from the sender for a long period of time. Standby timer t_{sb} is used in this situation.

The timer counts the time of waiting for the next data packet or Heart Beat packet transmitted over the connection. On the timer expiration the transport connection should be closed (Fig. 7.).

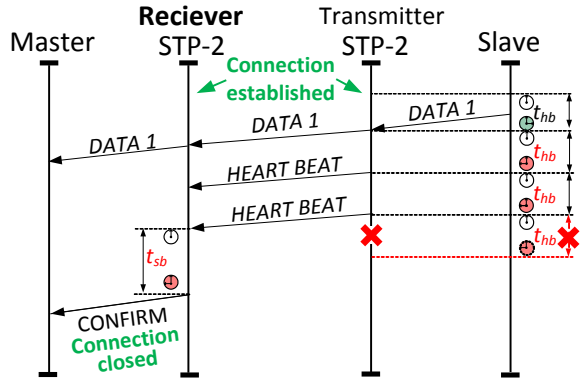


Fig. 6. The transport connection closure on standby timer expiration on the receiver side

Similar situation occurs on the transmitter side. If Heart Beat packets are not received after the timer expiration then transmitter should close connection (Fig. 7.).

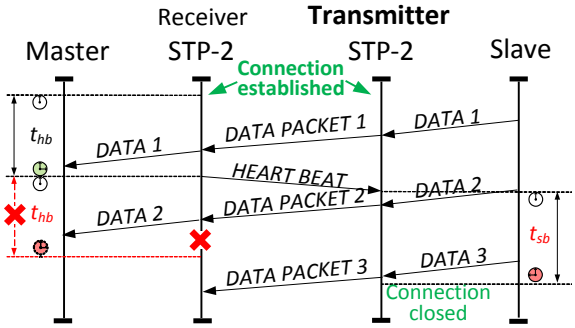


Fig. 7. The Transport connection closure on standby timer expiration on the transmitter side

F. Data transmission phase

STP-2 provides data transmission in two directions for the first operating scheme: from the master and from the slave device. Data are transmitted between two slaves in the second operating scheme. Transmission direction is specified in the OCR packet. Data packets are sent only after transport connection establishment (see Fig. 8. , Fig. 9.).

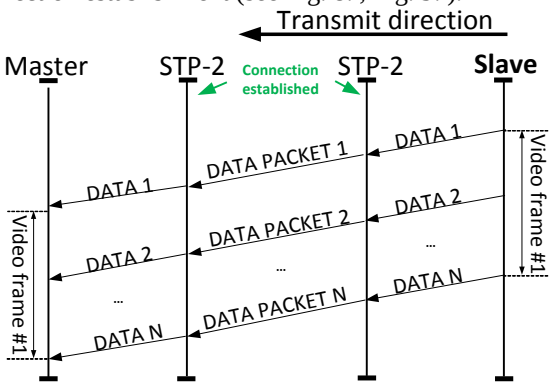


Fig. 8. Data transmission from the slave device

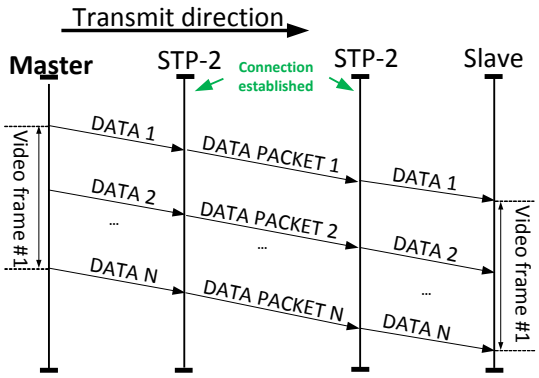


Fig. 9. Data transmission from the master device

In STP-2 data from the application (for example, an onboard spacecraft camera) should be sent N-Char-by-N-Char. (The bit width of interface with application can vary in different implementations).

Data flow is divided into PDUs; Every PDU together with the data packet header and tailer is sent through the SpaceFibre network.

When the first operating scheme is used (Fig. 2) if the transmitter has no data to send, then STP-2 should issue Heart beat service packets. The HeartBeat is addressed to the second device (master or slave). The packet notifies remote device that implementation of STP-2 protocol on the transmitter side is in the state of operability, but there is no data from application to send. Also the receiver should send Heart Beat packet to indicate that it is serviceable.

When the second operating scheme is used (Fig. 3) the master sends the Heartbeats to the both slaves and every slave sends HeartBeats to the master.

If the slave does not reseive a HeartBeat due long time it should close connection. (If the master is fail, other (spare) master can open connection with this slave due this mechanizm.)

G. Data Size Control on the Transmitter Device

STP-2 guarantees that the size of transmitted data of every data packets over established connection will not exceed the parameter of max accepted data size. The parameter is specified during the connection establishment. For this purpose transmitter should count the number of sent bytes. If the counter has reached the value of parameter, then the packet transmission should be terminated with EEP; the STP-2 controller should write information about this error to status register.

H. Control of Packet Sending Frequency at the Transmitter Device

STP-2 provides control of data packet sending frequency in order to maintain constant transmission rate. There is the packet send timer on the sender side. Timer duration is configured via an OCR packet. Only one data packet is allowed to be sent during the timer period. If the timer expired and End-of-Message symbol was not received from the application, then STP-2 should complete packet transmission by inserting cut-

error indication into “Sending flag” field of data packet. The field is the last byte of data packet before EOP. STP-2 should write information about this error in a status register.

I. Data Size Control at the Receiver Device

STP-2 guarantees that the size of received data of every data packets packet over established connection will not exceed the parameter of max accepted data size. The parameter is specified during the connection establishment. Receiver (like transmitter) should count the number of received bytes. Counter should be incremented only to amount of data passed to applications (for example, monitor). If the counter has reached the parameter value, reception of packets should be terminated. The rest of packet bytes should be discarded till reception EOP or EEP. STP-2 should notify application about it when last payload bytes will be transferred to Application layer. Information of the occurred error should be written to STP-2 status register.

J. Packet Filtering on the Receiver Device

STP-2 detects errors in the packet header and payload at the receiver side. Received packets with a CRC header error should be discarded as incorrect. If a CRC error of the payload is detected then this packet should be passed to the applications with indication of the error. STP-2 should write information about this error to status register.

K. Out-of-order Data Packet Detection

STP-2 supports out-of-order data packet detection. For this purpose data packets are numbered by time stamp on the transmitter side (Fig. 10.). Time stamps are independent from Reset actions. It is monotonically increased.

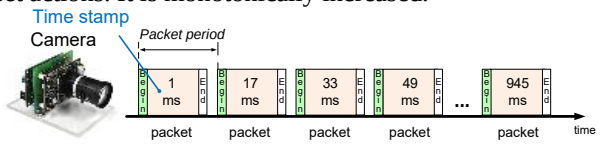


Fig. 10. Data Packets numbering by time stamps

L. Statistics report

STP-2 allows monitoring of streaming data delivery. It provides feedback on the quality of service in streaming data distribution by periodically (or by request) sending statistics information to master device in a transport connection. For this purpose Status service packets are issued by the receiver or/and the transmitter. The Packet includes the following information:

- error code (packet loss, incorrect packet size, packet delay variation, etc.);
- numbers of such errors (errors of such type);
- occurrence time (first and last time).

An application, that has received such information, may control quality of service parameters, perhaps by limiting flow.

VII. Conclussion

The paper gave an overview of actual problem to deliver streaming video over onboard spacecraft SpaceFibre networks. Streaming traffic features and streaming video characteristics were considered. Industrial standards for streaming video transmission ARINC-818-2 and CCSDS Digital Motion Imagery were analyzed. Main requirements for streaming protocols were presented. The overview of existing streaming protocols was done.

In this article a new STP-2 protocol for streaming data over onboard SpaceFibre networks was proposed. This protocol is based on STP protocol. STP is adapted for streaming data transfer. STP-2 has some important modifications which allow to improve delivery of streaming data flows over high-rate SpaceFibre networks. They are: two operating schemes are suitable for onboard spacecraft systems (master-slave and master-two-slaves), quick transport connection establishment and closure, data size control on the transmitter and receiver devices, stable intensity of streaming data delivery, packet filtering, out-of-order packets detection, monitoring of streaming data delivery and state of device’s operability.

Acknowledgment

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under the contract 1810 in 2016.

References

[1] E. Mikrin, Design concept of Spacecraft Onboard Control System. Moscow: Bauman MSTU Publ., 2003, 336 p. (In Russian)

[2] I. Korobkov, “Adaptive Data Streaming Service for Onboard Spacecraft Networks” Proceedings of SUAI Scientific session, Saint-Petersburg State University of Aerospace Instrumentation (SUAI), Saint-Petersburg, SUAI, 2015, pp. 73-82. (In Russian).

[3] Y. Sheynin, E. Suvorova, I. Korobkov, J. Petrichkovich, T. Solokhina, A. Glushkov, A. Sakharov, “Intelligent Networking for Distributed High-Rate Streaming Sensor Fields,” in Proceedings of 16th edition of Sophia Antipolis MicroElectronics (SAME) Forum 2013. Sophia Antipolis, 2013.

[4] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements, SpaceFibre Specification, Draft F3, September 2013, 161 p.

[5] ARINC Inc., ARINC Specification 818-2 (ARINC 818 Supplement 2), 2013. 131 p.

[6] CCSDS Digital Motion Imagery Standard 766.1-B-1, 2015. 41 p.

[7] Order No. 113 dated 27.09.2007 "On approval of Requirements to organizational and technical ensuring of sustainable functioning of the communication network", the Ministry of communications, 2007, 5 p. (In Russian)

[8] GOST R ISO/TR 16056-1-2009. Health informatics. Interoperability of telehealth systems and networks. Part 1. Introduction and definition. Moscow, Standartinform Publ., 2011, 10 p. (In Russian).

[9] N. Olifer, V. Olifer, Computer networks: Principles, Technologies and Protocols for Network Design. Wiley, 2012.

[10] Ali C. Begen, T. Stockhammer, HTTP Adaptive Streaming: Principles, Ongoing Research and Standards, Cisco, 2013

[11] F. L. Schiavoni, Possibilities in Network Transport Protocols to Audio Stream Application Context, Institute of Mathematics and Statistics, University of Sao Paulo, 2011.

[12] RGB Networks, Comparing Adaptive HTTP Streaming Technologies, 2011, 20 p.

[13] Mosleh M. Abu-Alhaj, Ahmed Manasrah, Mahmoud Baklizi, Nibras Abdullah, Lingeswari V. Chandra, Transport layer protocols taxonomy from Voice over IP perspective. Advanced Computing: An International Journal, Vol.2, No.4, July 2011.

[14] I. Korobkov, "Adaptive Data Streaming Service for Onboard Spacecraft Networks," in Proceedings of 17th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program. P.G. Demidov Yaroslavl State University, Yaroslavl, Russia, 2015, pp. 291-298

[15] Y. Sheynin, E. Suvorova, F. Schutenko, V. Goussev, "Streaming Transport Protocols for SpaceWire Networks," International SpaceWire Conference 2010, Saint Petersburg, 2010.

Deterministic Services for SpaceWire Networks

SpaceWire networks and protocols, Long Paper

Valentin Olenev, Elena Podgornova, Irina Lavrovskaya, Yuriy Sheynin
Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
{valentin.olenev, alena.podgornova, irina.lavrovskaya}@guap.ru, sheynin@aanet.ru

***Abstract*—Deterministic behavior is an important paradigm for verification and validation of real-time systems such as those on crewed space vehicles and robotic spacecrafts. Providing deterministic characteristics of the data transfer for the spacecraft that uses the SpaceWire technology is an essential problem, especially for autonomous vehicles like satellites. Deterministic data delivery guarantees that transmission of data from one node of the onboard network to the target node would not take longer than the specified time period. Such task is solved by using specific communication protocols that include a scheduling service.**

Modern space industry demands a protocol running over SpaceWire, which can provide deterministic data transmission characteristics. The scheduling problem becomes more complicated, when we consider a number of communication protocols simultaneously operating in every node of the network, e.g. RMAP, STP-ISS, CCSDS PTP. Traffic from different transport protocols can interfere especially while getting access to the SpaceWire link in a node.

The paper presents Multiprotocol Scheduling Service - a new scheduling protocol for SpaceWire networks which provides deterministic data delivery in a network and performs arbitration of data coming from several transport protocols. Firstly, we give an overview of TDMA-based network protocols that have been developed for the ground-based and onboard networks. Then, we present Multiprotocol Scheduling Service which is based on the STP-ISS scheduling mechanism and extended with additional features.

***Index Terms*— Scheduling, Determinism, SpaceWire, STP-ISS, On-board Network, Quality of Service.**

I. Introduction

Determinism is a philosophical doctrine stating that all events are caused by things that happened before them and that people have no real ability to make choices or control what happens. The same is for the behavior of the complex systems and networks. Deterministic systems have predictable behavior, which is necessary to perform analysis to ensure requirements are met. Deterministic data delivery guarantees that data from one node of the onboard network would be delivered to the target node in a fixed time. The developer can schedule all the onboard traffic and prevent the potential deadlock and data delivery delays. Such task is solved by using a specific communication protocol that includes a scheduling service.

If we have a number of different entities that transfer data to the network, we need a single scheduling instance that would be able to control all the data transfer from the particular node. Deterministic characteristics are obtained by using time-division multiplexing (TDMA). Time-division multiplexing (or scheduling) is used in network technologies to obtain guaranteed latency and throughput for user data, and to avoid conflicts with simultaneous network resources usage.

Current paper provides an overview of existing communication protocols that use scheduling quality of service and compares them. Also we propose a solution for scheduling of traffic from several transport protocols and applications operating on top of SpaceWire. It is a new Multiprotocol Scheduling Service (MSS) which gives different options to guarantee that data in the SpaceWire network would be delivered to the target in time.

II. Examples of TDMA-Based Deterministic Data Delivery Protocols

Time-division multiplexing problem is an important issue for communication protocols where deterministic data delivery is required. Time multiplexing is actively used in 2G and 3G mobile networks, as well as in some wireless personal networks, such as Bluetooth, ZigBee, Ubiquiti.

SpaceWire on-board networks also require time-division multiplexing solutions as the technology that is used in spacecraft and avionics. We reviewed a number of ground-based and on-board network protocols. Current section provides an overview of these protocols with scheduling quality of service [1], [2], [3]. This will help to understand, which mechanisms and algorithms are used to provide deterministic data delivery for different tasks.

Time multiplexing requires periodic synchronization between nodes. For this purpose, the protocol can use specific messages from the time-master to synchronize local clock. For example, there are reference messages in TTCAN [4], [5] which are transmitted in every basic cycle. Profinet IO IRT [6] implements Precision Transparent Clock Protocol (PTCP) [7]. PTCP synchronizes the clock of the network nodes by periodical

broadcasting of the synchronization frames, which are sent in every communication cycle. SpaceWire-D [8] and STP-ISS [9], [10] use time-codes, which are sent in every time-slot and epoch respectively. TTEthernet [11], [12] defines Protocol Control Frames (PCF) which contains accumulated time information regarding its passing from sender to a receiver. PCF should be sent once in the cluster cycle.

There are some interesting mechanisms used by other protocols. In TTP/C [13], for example, a node can calculate the difference between the clock of the sending node and its own clock by noting the time when messages are received from other nodes with the known schedule (TTP/C is a broadcast protocol, so all nodes receive all messages). Flexray [14, 15], uses similar approach: every node in each channel shall measure and store the time differences between the expected and the observed arrival times of all sync frames received during the static segment, calculate and apply clock correction term during the network idle time. Byteflight [16] nodes synchronize on cyclical synchronization pulses generated by SYNC master in every time interval. In SpaceFibre [17] the time value shall be taken from the local time register, which is regularly updated by the time-distribution broadcast channels. TSN [18] uses physical layer timestamps to compute network delays and define synchronization events.

Some protocols achieve determinism by using scheduling not only in nodes but in switches also. For example TSN uses concept known as the "time-aware shaper" (TAS), which deterministically schedules traffic in queues through switched networks. With the time-aware shaper concept it is possible to control the flow of queued traffic from a TSN enabled switch. Ethernet frames are identified and assigned to queues based on the priority field of the virtual local area network (VLAN) tag. Each queue is defined within a schedule, and the transmission of messages in these queues is then executed at the egress ports during the scheduled time windows. Other queues will typically be blocked from transmission during these time windows, therefore removing the chance of scheduled traffic being impeded by non-scheduled traffic.

In addition to a schedule for the network a protocol can use priorities. In SpaceFibre several virtual channels can be scheduled to send data in the same time-slot. In this situation medium access controller sends data from the virtual channel with the highest precedence.

Protocol can allocate only a part of the epoch for the scheduled traffic. For example, in Profinet IO the IRT part of the communication cycle is reserved for real-time communications, in which the deterministic message frames are sent.

Among all of the overviewed protocols only two are able to operate in SpaceWire networks: SpaceWire-D and STP-ISS. However, SpaceWire-D uses another transport protocol, RMAP, for transmitting data over the network, so it complicates the protocol hierarchy. Building any transport protocol over another transport protocol – the RMAP transport protocol, tangles the protocol stack and introduces unnecessary overheads.

Comparative analysis of different TDMA-based protocols features is given in Table I.

TABLE I. COMPARATIVE ANALYSIS OF DIFFERENT TDMA-BASED PROTOCOLS

Feature / Protocol	Topology	Synchronization	Static scheduling	Dynamic scheduling	Operation on top of SpaceWire	Synchronization period
TTCAN	bus	yes	yes	no	no	Once an epoch
Byteflight	bus	yes	yes	no	no	Once a time-slot
Flexray	bus	no	yes	yes	no	Once an epoch
TTP/C	bus	no	yes	no	no	Once a time-slot
TTEthernet	distributed	yes	yes	yes	no	Once an epoch
SpaceFibre	distributed	separate channel	yes	no	no	Undefined
SpaceWire-D	distributed	yes	yes	no	with RMAP	Once a time-slot
TSN	distributed	yes	yes	no	no	Implem. dependent
Profinet IO IRT	distributed	yes	yes	yes	no	Once an epoch
STP-ISS	distributed	yes	yes	no	yes	Once an epoch

III. MULTIPROTOCOL SCHEDULING SERVICE

Modern space industry demands a protocol running over SpaceWire, which can provide deterministic data transmission characteristics [2]. The basic SpaceWire standard covers three bottom layers of the OSI model and does not provide transport services [19]. Nowadays, there is a number of transport protocols intended to operate over SpaceWire. They are: RMAP, CCSDS PTP, STP-ISS, STUP, JRDDP, SpaceWire-R, STP and SpaceWire-D. Each of them is intended to solve its particular tasks and, in many cases, there are two or more transport protocols operating simultaneously in one network. Moreover, a single node can implement several transport protocols running over SpaceWire (for example, RMAP, CCSDS PTP, STP-ISS). Traffic from different transport protocols can interfere especially while getting access to the SpaceWire link in a node. It is rather difficult to avoid conflicts with simultaneous network resource usage, thus we cannot provide deterministic delivery of data in the network.

Consequently, there is not only an issue of schedule creation and synchronization between the nodes but also an issue of arbitration of different transport protocols' data flows. For SpaceWire networks only SpaceWire-D protocol deals with an

issue of scheduling in SpaceWire networks. However, SpaceWire-D was not designed for scheduling traffic from several transport protocols as it gets data directly from the Application Layer. Moreover, SpaceWire-D utilizes RMAP to communicate over the network, which imposes restrictions and effects on its flexibility in use.

Therefore, it was decided to take the STP-ISS scheduling mechanism as a basis and develop a new scheduling protocol, which solves abovementioned problems. Also we extended functionality of this mechanism by some abilities and features of other scheduling protocols. The new scheduling protocol is called Multiprotocol Scheduling Service (MSS). Fig. 1 shows the place of the MSS in the protocol stack and its comparison with the OSI reference model [20].

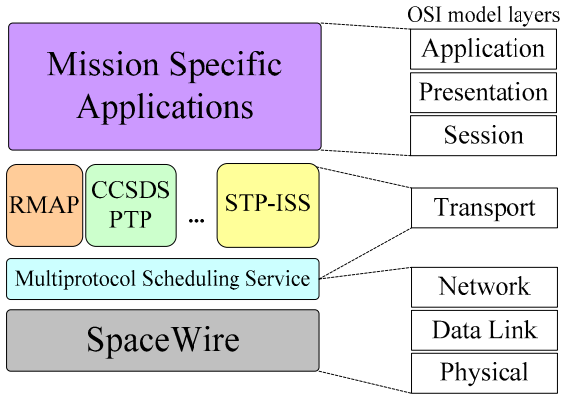


Fig. 1. Comparison of the MSS with OSI

The STP-ISS scheduling mechanism is based on SpaceWire time-codes distribution provided by the SpaceWire standard.

A. SpaceWire Time Synchronization

The local time ticks in each node must be periodically resynchronized within the global time base [9]. The STP-ISS rev.2 scheduling mechanism is based on the SpaceWire time-codes broadcasting mechanism. These time-codes contain a six-bit value of system time. Each node and switch has its internal six-bit time counter. There should be a single node or switch in a network, which is set as the time-master. It is responsible for time distribution over a network. When the time master receives a tick from a host-system, it should increment its time counter and send new time value in a time-code. When a node or a switch receives a time-code, it should update its internal time counter with the received time value. This new value should be one more than the time-counter's previous time value.. When a switch receives a time-code with time value, which is one more than the internal counter's value it increments the counter value and emits a tick signal. This tick signal propagates to all the output ports of the switch so that they emit the time-code. When switch receives a time-code with a time-code value that is equal to the internal counter value, then it is ignored. It helps to prevent circular time-codes propagation. This is the way the time-codes are used to synchronize all the network nodes with the time master's clock [19].

B. The STP-ISS Scheduling Mechanism

According to the STP-ISS scheduling mechanism, there is a single schedule for the whole SpaceWire network. It gives an opportunity for the node to send data only during particular time-slots. The schedule and time-slot duration are set during the configuration phase and are stored in each end-node of the network. The time-slot timer (T_{TS}) counts duration of the current time-slot for a particular node. Synchronization according to a scheduling mechanism is performed once in an epoch. An epoch has a constant number of time-slots. For example, an epoch can consist of 10, 20, 64 or more time-slots, but it should contain at least 2 time-slots. The scheduling table describes one epoch.

The number of time-slots in one epoch should be defined during the configuration phase and should be set to the time-slots counter C_{TS} value. The time-slot duration DTS should be set to the time-slot timer T_{TS}.

The epoch duration D_E is calculated in the following way:

$$D_E / D_{TS} \rightarrow C_{TS} \qquad (1)$$

If the time-slot duration D_{TS} value changes, the epoch duration value D_E should be calculated and updated.

Each node is permitted to send packets at a particular time-slot in accordance with the schedule. At the end of its time-slot, the node should stop the data transmission. However, the transmission actually stops only after the current packet is transmitted to the network (the STP-ISS protocol has limited PDU length). If any other node has data for transmission, but it is not scheduled for transmission at the current time-slot, then this node should wait for its time-slot.

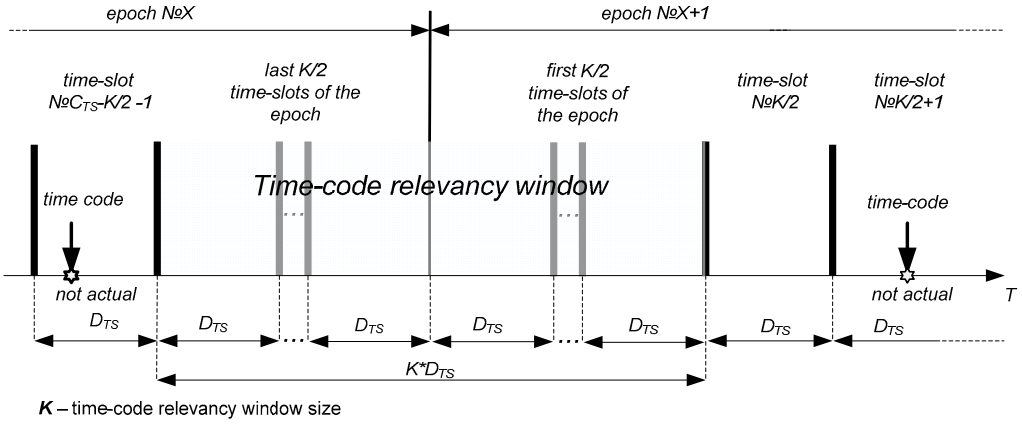


Fig. 2. Time-code relevancy window

STP-ISS protocol defines the time-code relevancy window that shows if the received time-code is relevant or not. This parameter defines a number of time-slots in the end of the epoch and in the beginning of the next epoch. During these time-slots a received time-code is considered as relevant (K is time-code relevancy window size). The time-code relevancy window is shown in Fig. 2. Time-code relevancy window is a configuration parameter and should be set during configuration phase. It could be defined individually for each node in accordance with the accuracy of local clocks.

The time-slot timer expiration in the last time-slot of an epoch and reception of a relevant time-code indicate beginning of the new epoch, in which the time-slot counter C_{TS} will count time-slots starting from zero. When the node gets the time-code, it does not analyze the time-code number. The beginning of a new epoch is associated with the fact of the time-code reception.

There are two possible synchronization cases, which can occur:

- the next time-code is received during first K/2 time-slots of the epoch;
- the next time-code is received during last K/2 time-slots of the epoch.

Considering the node functionality, the abovementioned cases mean that the internal time-slot timer and the time master are not synchronized. This means that the node should start the synchronization process.

Fig. 3 shows the case, when a node started a new epoch and the expected time-code is received during first *K/2* time-slots of the new epoch. In this case, the node should terminate time-slot timer T_{TS} and calculate new value for the time-slot duration. The D_{TS_new} value is calculated according to the equation (2):

$$D_{TS\_new} = D_{TS} + \frac{\Delta t}{C_{TS}}, \quad (2)$$

where Δt is the current value counted since the beginning of the epoch.

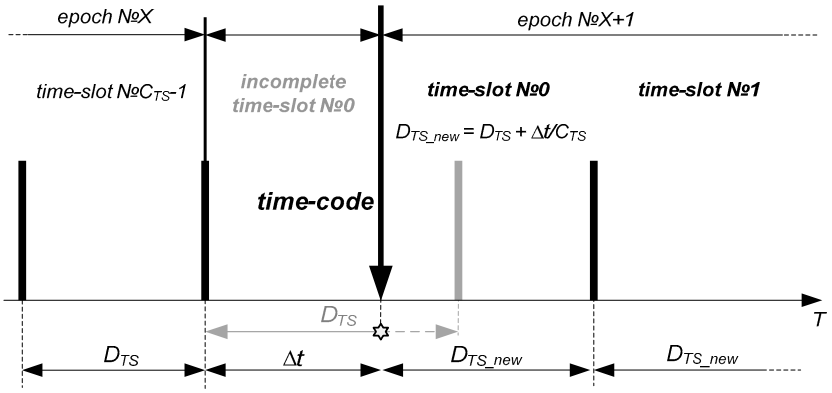


Fig. 3. Time-slot timer value correction (the time-code received during the first time-slot of the epoch

Subsequently, the node updates the epoch duration value according to the equation (3).

$$D_E = D_{TS\_new} \cdot C_{TS} \quad (3)$$

The newly calculated value will be applied to the T_{TS} timer for the next time-slot.

Let us consider the second case when the time-code is received during the last *K/2* time-slots of the epoch (see Fig. 4). In this case, the node should terminate the current epoch and calculate new values for the time-slot timer. For this purpose, the node takes the current Δt value counted since the beginning of the epoch and calculates the new time-slot duration according to the equation (4):

$$D_{TS\_new} = \frac{\Delta t}{C_{TS}} \quad (4)$$

The next time-slot starts with the new T_{TS} timer value D_{TS_new}.

If the epoch timer expires simultaneously with the time-code reception, then there is no need to correct the epoch timer value. The moment of the epoch timer expiration and the time-code reception is determined depending on the implementation. These events are not strictly simultaneous in the hardware. So there is some gap between these events that could be considered as satisfactory or not. Also this gap can be useful to take into consideration the accuracy or jitter of the time-code reception.

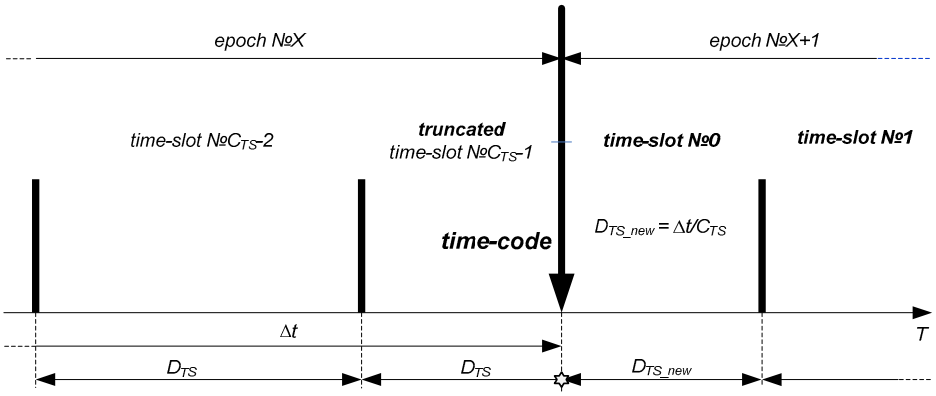


Fig. 4. Time-slot timer value correction (the time-code received the last time-slot of the epoch)

The STP-ISS protocol should count the number of received irrelevant time-codes. Reception of three irrelevant time-codes means that the internal time-slot timer and the time master are significantly asynchronous. In this case, it is necessary to synchronize with the time master. Reception of the third irrelevant time-code should determine the beginning of the new epoch. The node should terminate the time-slot timer and wait for reception of the next time-code. In this new epoch the node should not send data until reception of the next time-code. After reception of a time-code the node should update the time-slot duration value and then continue data transmission according to the schedule. New time-slot duration value should be calculated according to the equation (5):

$$D_{TS\_new} = \frac{\Delta t}{C_{TS}}, \quad (5)$$

where Δt is the time value, that is counted starting from the moment of third irrelevant time-code reception and finishing with the next time-code reception.

C. Multiprotocol Scheduling Service

The described above STP-ISS scheduling mechanism, has been extended with additional features for operation with several transport protocols simultaneously and results in a new scheduling protocol for SpaceWire networks - Multiprotocol Scheduling Service. It should be implemented not only in the end-nodes (as it is designed in STP-ISS) but also in switches which are directly connected to the nodes. Let us now consider MSS mechanisms and features in details.

1) Scheduling table: Generally, there is a schedule for the whole SpaceWire network defining in which time-slots a particular node can send data. An example of such scheduling table is given in Table II. Time synchronization mechanism was taken from the STP-ISS without any modifications. Scheduling table should be stored in each node and should be designed in such a way that an access to shared resources (e.g. output switch ports, links, etc.) is multiplexed in time.

According to the MSS a node can only send data in the specified time-slots. However, if there are several transport protocols operating in one node it is necessary to somehow share this time-slot between them. For this purpose we proposed to allocate full time-slot for a particular transport protocol, and if there is more than one transport protocol in a node scheduled to send data at one time – these protocols should be arbitrated by priorities.

TABLE II. SCHEDULING TABLE FOR THE NETWORK

Node	Time-slots																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0				■	■		■			■			■			■			■		■
1	■		■			■		■			■			■			■			■	
2		■			■				■	■			■		■	■			■		
⋮																					
25		■	■			■			■	■					■	■				■	

According to the proposed approach, an entire time-slot is assigned for transmission of data of a particular transport protocol. It gives an opportunity for the certain transport protocol on the certain node to send data only during a particular time-slot. In such case all other protocols, which operate in the node, should wait for their time-slots. An example of the scheduling table for this approach is shown in the Table III.

TABLE III. SCHEDULING TABLE FOR NODES AND TRANSPORT PROTOCOLS

Node	Protocol	Time-slot																				
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	STP-ISS										■						■					
	RMAP					■								■						■		
	PTP				■			■														■
1	STP-ISS	■							■						■							
	RMAP											■						■				
	PTP			■			■														■	
2	STP-ISS		■							■						■						
	RMAP									■							■					
	PTP					■								■						■		
⋮																						
25	STP-ISS			■			■														■	
	RMAP		■							■						■						
	PTP										■						■					

If several transport protocols are scheduled to send data at one time and have data for sending in the allowed time-slot, then the protocol with the higher priority will send data first. This gives an opportunity to effectively utilize the network capacity. Using of priorities is optional, so this mechanism could be switched off (or even not implemented in a node). In this case transport protocols scheduled for the same time-slot should send

data in the same order as they were prepared for the transmission.

The Multiprotocol Scheduling Service does not require any data buffers: it passes transport protocol data directly to SpaceWire without keeping it inside the scheduler. In the allowed time-slot the scheduler arbiter analyses whether transport protocols have data to send or not and makes a decision which transport protocol may transmit data. Then, the particular transport protocol can pass its data packet to the Multiprotocol Scheduling Service for sending over the network.

If the packet length is too big for the duration of the time-slot and we do not have enough time to send the packet, than we have two options:

- let the MSS finish the packet transmission, exceeding the time-slot boundaries;
- cut the packet and insert EEP in the end.

It is the configuration parameter of the scheduling protocol.

2) Transport protocols identification: Although each transport protocol has its own protocol identifier, it is difficult to identify what protocol or application sends the data to the network. This is due to using SpaceWire path addressing in the packets as it is not limited in length. Moreover, there could be applications, which do not have any protocol identifier but may send data directly to the SpaceWire without using any transport protocol.

Therefore, for the purpose of identification, each entity sending or receiving data passing through the MSS has its own SpaceWire logical address. In addition, the Multiprotocol Scheduling Service has a separate SAP for each protocol or application that is trying to send data by its means. Each SAP is associated with the logical address of the SAP service user.

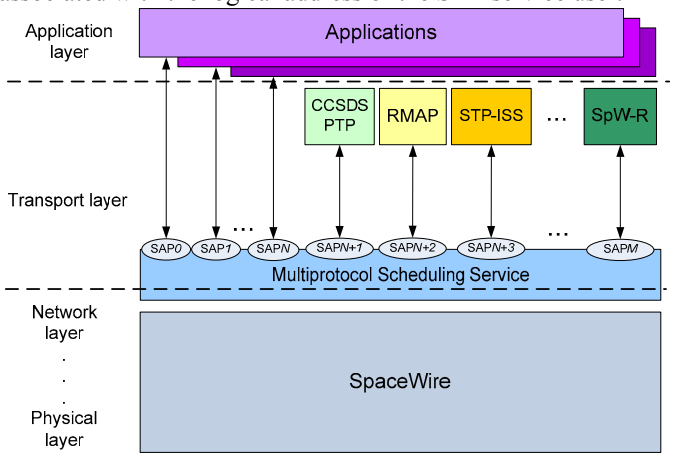


Fig. 5. Multiprotocol Scheduling Service SAPs with upper entities

If a transport protocol works with a number of applications on top, and each application has its own logical address, then SAP could be associated with a range of logical addresses. Applications also could send data directly to the scheduling protocol; so they also have separate logical addresses. An example of protocol and application identification by the MSS is shown in Fig.6.

If the Multiprotocol Scheduling Service gets a time-code or interrupt, it sends it to all the SAPs. It means that all transport protocols and applications have a possibility to get a control code, and if they need it – receive and process it. Also if a node with Multiprotocol Scheduling Service is a Time Master, then MSS should have an ability to transmit time-codes to the network and to all the SAPs of the protocol simultaneously. It is done to give ability to other applications or transport protocols in a node to get the correct time information.

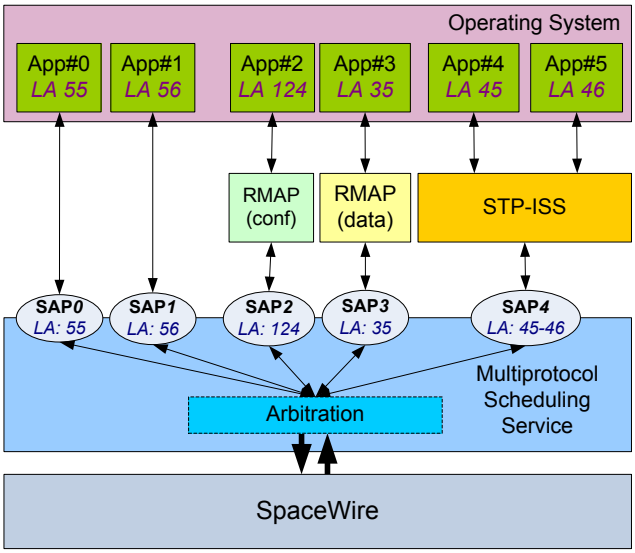


Fig. 6. Example of protocol and application identification

3) Switches: SpaceWire-D protocol is intended to be implemented in the SpaceWire nodes only, but the Multiprotocol Scheduling Service would be implemented in SpaceWire switches also to get guaranteed deterministic network behavior in any network configuration. In MSS we propose to use the port guardian mechanism, which is described in [1]. The following mechanism should be implemented in switches, which are directly connected to the nodes. This mechanism is optional and it can be switched off for a particular switch.

Most of described above network technologies suppose port guardian mechanism in order to protect the network from nodes that try to transmit data at inappropriate time-slots. Often such a "watch dog" is implemented as a separate device or chip in order to increase fault tolerance. The Port guardian guarantees that the node would not transmit data during wrong time-slots and eliminates «babbling idiot» problem.

SpaceWire switches store the scheduling table (see Table II), but it is simpler than for the nodes (Table III). Switch does not know which protocol sends the packet, but it should be able to block the packet transmission, if the node is not scheduled for data transmission in the current time-slot. The switch's scheduling table identifies, which port is allowed to send data during a specific time-slot. Thus, according to this schedule, the switch can determine whether the packet should be discarded or it can be routed to the outgoing port. If the leading byte of the SpaceWire packet has been received in the time-slot, in which the node is allowed to send data, then this packet can pass through the router. Otherwise, this packet should be discarded.

The Fig. 7 shows a SpaceWire network with network switches, marked as «Net guard», which store a scheduling table and permit data transmission. The switch, which is not connected to nodes, could be a standard SpaceWire routing switch, which is not able to analyze the scheduling.

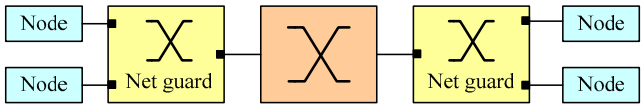


Fig. 7. Network with network guardians

2) Out-of-schedule nodes: If in a network there is a number of nodes that work out-of-schedule and are not included into a scheduling table, then it is better to isolate these nodes in a separate network region. The Switch that would connect this region to another, scheduled part of network should contain a scheduling table that will regulate data transmission from this region. An example of out-of-schedule region is shown in Fig. 8.

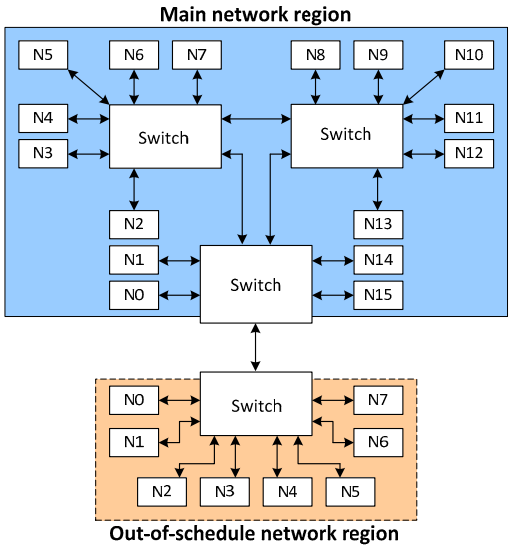


Fig. 8. Example of out-of-schedule region

IV. Conclusion

In the current paper we proposed new scheduling service that will provide determinism to SpaceWire networks, where multiple transport protocols and applications are operating. For this purpose we overviewed communication protocols, which use Time-Division Multiplexing (scheduling) concept to provide the deterministic data delivery. We analyzed the scheduling mechanisms and decided to take the STP-ISS scheduling mechanism as the basis for a new service. Also we proposed some improvements and additions to this mechanism that increase the quality of deterministic data delivery. This mechanism can prevent network resources usage conflicts and increase the network bandwidth.

The proposed Multiprotocol Scheduling Service (MSS) will solve the serious problem of deterministic data delivery that currently is one of most important tasks for space industry.

Acknowledgement

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under the contract RFMEFI57814X0022.

References

[1] I. Korobkov, E. Podgornova, D. Raszhivin, V. Olenev, I. Lavrovskaya "Scheduling Mechanisms for SpaceWire Networks", *Proceedings of 16th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program*; Russia, Yaroslavl, 2015. pp. 82-88.

[2] V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov, "Analysis of the Transport Protocol Requirements for the SpaceWire On-board Networks of Spacecrafts", *Proceedings of 15th Seminar of Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program*; Saint-Petersburg: Saint-Petersburg University of Aerospace Instrumentation, 2014. pp. 65-71.

[3] S. Balandin, A. Heiner, "Dynamic Localized Load Balancing", *SPIE Volume 5244: Performance and Control of Next-Generation Communications Networks (ITCom2003)*, USA, September 2003, pp. 164-175.

[4] Fuehrer, T., Mueller, B., Hartwich, F., and Hugel, R., "Time Triggered Communication on CAN (Time Triggered CAN-TTCAN)," SAE Technical Paper 2001-01-0073, 2001.

[5] F. Hartwich, T. Führer, R. Hugel, B. Müller, Robert Bosch GmbH, "Timing in the TTCAN Network"; in Proc. of the International CAN Conference; 2002.

[6] S. Parkes and A. Ferrer-Florit, *SpaceWire-D – Deterministic Control and Data Delivery Over SpaceWire Networks*, Draft B. April 2010.

[7] D. Fontanelli, D. Macii, S. Rinaldi, P. Ferrari, and A. Flammini, "Performance analysis of a clock state estimator for Profinet IO IRT synchronization,", *Instrumentation and Measurement Technology Conference (I2MTC)*, 2013 IEEE International, May 2013, pp. 1828-1833.

[8] S. Parkes, A. Ferrer-Florit, A. Gonzalez. and C. McClements - *SpaceFibre Draft H1*: Space Technology Centre, University of Dundee, 2013.

[9] Z. Hanzalek, P. Burget; P. Sucha, "Profinet IO IRT Message Scheduling With Temporal Constraints", *IEEE Transactions on Industrial Informatics*, 2010, pp. 369-380

[10] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The Time-Triggered Ethernet (TTE) Design," *in Proc. 8th IEEE International Symposium on Object-oriented Realtime distributed Computing (ISORC)*, Seattle, 2005.

[11] TTTech Computertechnik AG, IEEE TSN (Time-Sensitive Networking): A Deterministic Ethernet Standard, Web: https://www.tttech.com/download/technologies/deterministic-ethernet/time-sensitive-networking/file/1c49796102083798d7f3968fa4c3c2ae3b59a471/.

[12] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov "STP-ISS Transport Protocol for Spacecraft On-board Networks", *Proceedings of 6th International SpaceWire Conference 2014 Program*; Greece, Athens, 2014. pp. 26-31.

[13] J. Berwanger, M. Peller, R. Griessbach, "Byteflight – A New Protocol for Safety Critical Applications" *in Proc. of the FISITA world Automotive Congress*, 2000.

[14] Flexray. FlexRay Communications System Protocol Specification Version 3.0.1. Specification, FlexRay Consortium, 2005.

[15] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, S. Kochura, S. Openko, D. Dymov "STP-ISS Transport Protocol Overview and Modeling", *in Proc. of 16th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program*; Oulu: University of Oulu, 2014. pp. 185-191.

[16] A. Hanzlik, “A Case Study of Clock Synchronization in FlexRay”, Research Report 31/2006 Technische Universitat Wien, Institut fur Technische Informatik, 2006.
[17] H. Kopetz, “Real-Time Systems. Design Principles for Distributed Embedded Applications”, Kluwer Academic Publishers, Boston, 1997.
[18] A. Ademaj, H. Kopetz, P. Grillinger, et al.,“Fault-Tolerant Time-Triggered Ethernet Configuration with Star Topology” *in Proc. 19th International Conference on Architecture of Computing Systems*, 2006.
[19] ESA (European Space Agency). Standard ECSS-E-50-12C, Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization. Noordwijk: ESA Publications Division ESTEC, 2008.
[20] A.S. Tanenbaum, *Computer Networks*, Fifth Edition; Prentice Hall, 2011.

Placement of Plug-and-Play Network Managers in SpaceWire Networks

SpaceWire networks and protocols, Long Paper

Ksenia Rozhdestvenskaya, Nadezhda Matveeva, Lev Kurbanov, Aleksey Evdokimov, Elena Suvorova, Yuriy Sheynin, Aleksey Rabin
Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russia
{ksenia.khramenkova, nadezhda.matveeva, lev.kurbanov, alexey.evdokimov}@guap.ru, {suvorova, sheynin}@aanet.ru, alexey.rabin@guap.ru

***Abstract*—Satellite onboard control systems have changed significantly towards of complexity of control algorithms. Network operation must be flexible and adapt easily to configuration and composition changes of the onboard network. Plug-and-Play should solve this problem. Plug-and-Play technology is aimed for network monitoring, should promptly detect changes in it and correctly handle any network situation. Due to size increase of onboard networks, one center (Plug-and-Play Manager), which would quickly react to changes in the entire network, is not enough, one must have several such centers. Thereby there is the problem of Plug-and-Play manager's placement in the network, to provide rapid response and efficient management of other devices in the network.**

Depending on the SpaceWire network functions and requirements, it must be taken into account in its design, in determining position for the Plug-and-Play managers. In the paper, we consider some reference variants of SpaceWire/GigaSpaceWire networks:

- **Network designer chooses the number of Plug-and-Play managers according to the network size and others parameters. The network is not clustered, not divided into subnetworks.**
- **Network designer chooses the number of Plug-and-Play managers according to the network size and others parameters. However, the network that is initially undivided into subnets, in the search for of the network managers location is divided into subnets.**
- **Network designer clusters the network, divides it into subnets. For each subnet, the manager's placement problem is solved separately in accordance with the requirements to particular subnet.**

In this paper, we will consider in details two approaches to determining location of Plug-and-Play centers in SpaceWire/GigaSpaceWire networks.

The first approach is based on a method of graph partitioning into subgraphs. The second approach is based on a P-median graph search algorithm. Both approaches are described in details. Advantages and disadvantages of these methods will be presented; their computational complexity will be evaluated.

In network managers placement it is important, what will be characteristics of data transfer from Plug-and-Play managers to nodes. These characteristics include data transmission time, data transmission delay, etc. It is important how the network nodes are distributed among the Plug-and-Play managers for control. The network managers loading should be as uniform as possible. All these parameters directly affect the quality of the Plug-and-Play technology.

We assess transmission characteristics between the Plug-and-Play manager and network nodes for each of these approaches, give recommendations of using these approaches for Plug-and-Play managers placement, taking into account various features of the construction and administration of SpaceWire/GigaSpaceWire networks.

***Index Terms*— onboard networks, network management, SpaceWire, GigaSpaceWire, Plug-and-Play.**

I. Introduction

Creation of modern onboard networks requires innovative approaches for its administration and management, among them - automation of all possible processes, configuration flexibility. PnP technology means network management without human configuring the onboard network. In the PnP technology, devices use a special algorithm, to research and configure each other. For this technology, one must have a network of some "smart" devices that will execute PnP algorithms.

Depending on the goals and tasks, a number of the required PnP managers is set, which must be present and be in operation in the network. In case of centralized PnP, there is only one manager in the network. This manager performs configuration and full network administration. However, for large networks with a large number of subscribers one manager may be not enough, because it cannot enough quickly manage the network. Therefore, a decentralized mode can be used, in which there are several managers. In the first and in the second case it is necessary to choose a place(s) of managers in the network structure.

For centralized mode, it is reasonable to locate manager in a center of the network.

Before choosing a place of managers' connection in a decentralized mode, the network should be divided into several regions.

In some projects, splitting onboard computing network of the satellite is done in the design phase, in creation of a network in accordance with its logical and physical structure. For example, the nodes and routers, which are located in one instrument area, may be defined as a separate region. In each such region a separate manager is used. It can be placed in the center of the subnet (just as a manager of the entire network is placed in a centralized mode).

However, in many other projects, developers do not carry out the network division into regions. In this case, a decentralized mode PnP technology is used; the network can be splitted into several regions. Then you need to select position of the manager connection for each region separately.

To solve the problem of network splitting into regions it is convenient to represent the network as a graph: the vertices of the graph correspond to the network devices, edges correspond to physical communication channels between devices. For easy way, we present non-oriented graph, as the data transfer can be in both directions for SpaceWire duplex communication channels. Weight of an edge between two vertices connected by this edge is a distance equal to one.

In another approach, the network originally isn't divided into separate regions - managers are placed in the network by using algorithm, which finds a P-median in the network (where P - is the number of placed managers).

To check the results of network's partition we use imitation modeling.

II. DIVISIONS OF A NETWORK INTO REGIONS APPROACH

A. Partition of a graph on subgraphs

The task of partitioning of a graph into subgraphs is deeply explored and there are different algorithms for its solution. Different algorithms are suitable for different requirements. The following requirements are relevant for our task: subgraphs that contained one node shan't be created; ability to control of subgraphs quantity; taking into account graph connectivity and closeness of a result to optimum (that is a result, which on these or those signs is more preferable than others).

To divide a graph into the required quantity of subgraphs, the following input data are necessary:

1) A specification of the graph structure.

2) Number of managers of the Plug and Play and their characteristic.

The manager of the Plug and Play is capable to support different number of devices (nodes of a graph) correspondingly to its technical characteristics, such as: memory size; performance; throughput and other. In some cases (the centralized Plug and Play) one manager of the Plug and Play is capable to service all given network entirely. The search algorithm of a median line of a graph is used for finding a network connection point for the manager position in the graph.

In case of the decentralized Plug and Play mode, before using the search algorithm of median line in a graph, it is necessary to divide the graph into subgraphs. The quantity of subgraphs depends on capabilities and resources of the selected Plug and Play managers.

The algorithm, which is capable to divide a graph into a given quantity of subgraphs taking into account connectivity, without formation of single subgraphs - multi-level algorithm of Kernighan-Lin [1]. It consists of three stages (Fig. 1):

1. The Stage of coarsening
2. The Separation stage
3. The Recovery stage

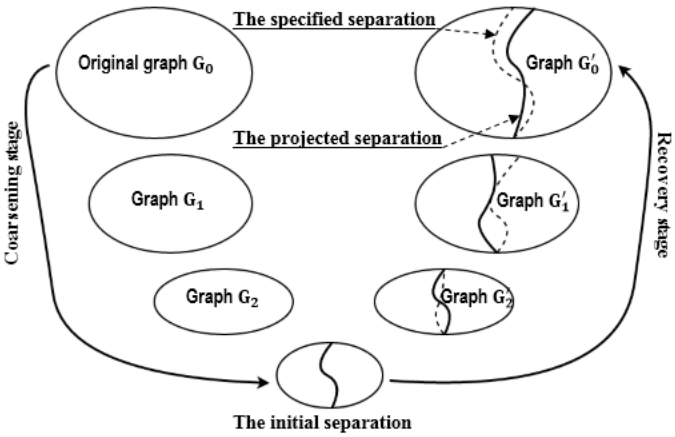


Fig. 1. Stage multilevel algorithm.

At the stage of coarsening a sequence of smaller graphs, each with fewer nodes, is constructed. Coarser graph can be obtained by tightening adjacent nodes. The edge between two nodes is deleted and the multinode is created, which consists of these two nodes. The algorithm "Random matching" was chosen to implement this stage. The random matching basic idea is that the nodes are visited in a random order. If the node u wasn't included in matching, then the algorithm randomly selects one of its adjacent nodes, which is also not included in matching. If such a node v exists, then the algorithm includes an edge (u, v) in matching; it also marks nodes u and v as visited. If there is no unmarked adjacent node v, then the node u remains free and passes into the following graph. This stage of coarsening is needed by the fact that an initial network can have big size. Adding this stage in the multi-level algorithm allows to reduce considerably an operating time of the separation algorithm for networks with large number of nodes.

The second stage of a multilevel algorithm – the separation stage. The Kernighan-Lin algorithm is selected for bisection of the graph. It begins with the initial division of the graph in half. At each iteration it searches for a subset of nodes from each part of the graph, such that the exchange of these subsets leads to separation of smaller cross-section. If such subsets exist, then the exchange is made, and this becomes the partition for the next iteration. The algorithm continues by repeating the entire process. If it can't find two such subsets, then the algorithm comes to an end because for this division the local minimum is reached, and any further improvement can't be made by the Kernighan-Lin algorithm. The Kernighan-Lin algorithm finds a local minimum for the separation, when it starts with a good initial separation. The requirement of repetitive calculations can be quite cumbersome, especially if

the graph is large. However, since the input of the separation algorithm receives a much smaller coarse graph, performing multiple requires very little time.

The last stage of the multilevel bisection – the recovery stage. During the recovery stage separation obtained for the coarser graph is projected back to the original graph. A more accurate graph has greater number of degrees of freedom that can be used to improve the separation and reduce the weight of the separator. For this reason, after division, the algorithm of division refinement is used. As the algorithm of refinement the described above Kernighan-Lin's algorithm is selected. The main idea of its application is in using the found solution for a coarse graph as an initial division for the Kernighan-Lin algorithm described at this stage. The reason is that this projected split is already good; therefore, Kernighan-Lin will converge within a few iterations to the best separation.

Thus, having passed the initial graph through all three stages, the output of the multilevel algorithm will be the two subgraphs that are most connected inside. To get more subgraphs one could put one of the resulting subgraphs to the input of the algorithm and repeat it.

After division of the graph into desired number of subgraphs, each of them, using a search algorithm median graph's is searched for a node, some centre to which a Plug-and-Play manager is connected. As mentioned above, this is a graph's median search algorithm [2].

In a number of tasks about placement of service stations it is required to locate a service station in the graph so that the amount of the shortest distances from this point to peaks of the graph was minimal. The optimum location of point in the specified sense is called the median line of a graph. Proceeding from the nature of target function, such tasks call minisummny tasks of placement [3]. The matrix of the lengths of the shortest paths between all nodes in the graph is calculated using the algorithm of Floyd–Warshell and used to determine the median graph, which is determined by the smallest value of the total distance from one node to all others [4].

After performing the graph partitioning and the center search in each subgraph the output data will be formed that contain:

1) A description of the network structure, with the distribution of devices across the regions.

2) A List of nodes to which Plug-and-Play managers are connected.

The presented algorithm is a heuristic, it is able to issue an acceptable solution in most practically important cases. However, in contrast to the exact algorithm for finding P-medians, it has the following features:

- doesn't guarantee finding of the best solution;
- doesn't guarantee finding of the solution even if it exists;
- can issue the incorrect solution in certain cases.

To identify the effectiveness of this algorithm in practice, we conducted a series of tests on the networks of different size and different topologies. a serial execution of the algorithm on networks was produced:

- Mesh 4x4 (16 nodes).
- Mesh 6x6 (36 nodes).
- Mesh 8x8 (64 nodes).
- Arbitrary topology (61 nodes).

For each network, the division into three regions was done 10 times. Among all the obtained solutions the best was found according to two criteria. The first is the minimum distance (in number of switches on the path) between the Manager and the most remote from it node in the subgraph. The second is the difference (in number of nodes) between the resulting subgraphs' sizes.

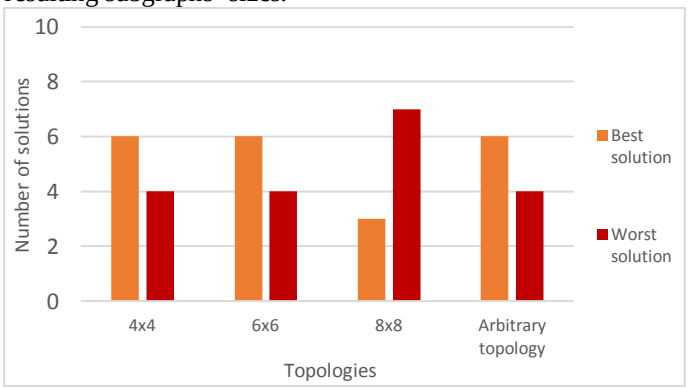


Fig. 2. The ratio of best and worst solutions in different topologies for a heuristic algorithm

The diagram in Fig. 2 illustrates a ratio between the number of the received partitions. The blue column shows how many solutions for a network with this topology are the best, and orange – the number of opposite solutions. Apparently from the diagram, the amount of the best results prevails on all networks, except a Mesh topology 8x8. As the reason for that serves the network size: as more nodes and edges are in the network, there are more options for graph division. Thus, the algorithm will issue more various solutions, thereby the number of the best solutions will make a smaller share of all possible. In case of arbitrary topology though the quantity of nodes is comparable with a Mesh topology 8x8, number of communications are less; the structure of a network too considerably influences the number of different solutions also.

The diagrams below (Fig. 3 and Fig. 4) provides a comparison of algorithm graph partitioning to subgraphs with the second, the exact algorithm considered in this article according to the selected criteria.

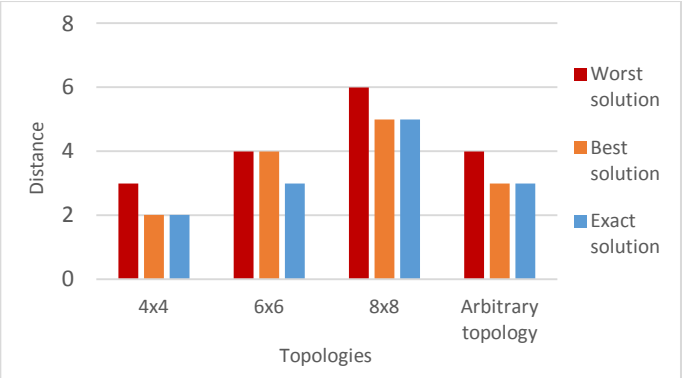


Fig. 3. The maximum distances in regions of the worst, best and exact solutions

Provided in Fig. 3 diagram allows to see a difference between the best solution (an orange column), the worst (red) and exact (blue) by the first criterion – the maximum distance between the manager and the most remote node in all regions. From the diagram it is visible that the algorithm is capable to issue a solution, comparable by this criterion with exact. The fact that for a Mesh 6x6 the exact solution turned out on 1 hop better says only that there was not enough number of the carried-out tests for receiving partition, comparable with exact.

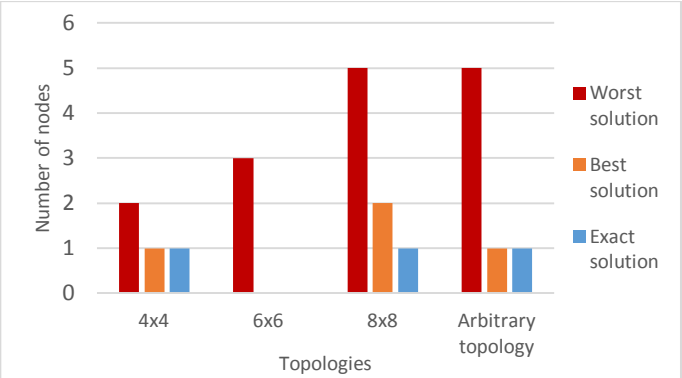


Fig. 4. Loads of managers of the worst, best and exact solutions

From the diagram in Fig. 4 one can see the difference between the best (orange column), worst (red) and exact solution (blue) according to the second criterion: the difference between the sizes of the obtained regions. As from the previous graph, it is clear that this algorithm allows to obtain a solution comparable with the exact by the second criterion. However, unlike the previous chart, the difference between the best and worst value for the second criterion is large.

A comparison of the solutions, obtained by the heuristic algorithm, with the exact, according to the selected criteria, but separately for every criteria, was above. If we compare the results under both criteria, it turns out that only for a network with arbitrary topology solutions are comparable to exact.

Thus, we can conclude that the algorithm gives best results for networks with arbitrary topology. It does not mean that for networks with a regular topology (Mesh) it is impossible to obtain a solution comparable to accurate. But for this solution it will be necessary to conduct a greater number of runs of the algorithm.

III. P-MEDIAN APPROACH

P-median problem can be often found in logistics [5,6]. This problem is from graph theory. In its classical form, it looks like: to place p services on the graph thereby the sum of the shortest distances from each services to the other graph vertices is the minimum possible. We have changed the classical form for solving the problem of placement managers in the network. In our case it is necessary to place p managers (p-median) in the network thereby the distance of each manager to the nearest nodes is as low as possible. The main condition for the solution existence is a full coverage of the vertex set V of the graph G (V, U) by union the median subset V_P (includes managers) and attachment subset V_A (includes nodes in the obtained regions) (1).

$$V_P \cup V_A = V \quad (1)$$

This approach finds managers location in the network, and then the network divides into regions. The region is defined by a set of nearest nodes from its manager. The distance between two nodes is the sum of edges weights in the path. Each edge has distance equal one.

P-medians algorithm is exact, and important feature of this algorithm is that any solution is the best of all possible solutions to the given criteria. There may be several such solutions.

The optimal solution criteria:

- The minimum distance from the manager to other nodes in the region. The minimum distance is necessary to reduce transmission delays between the manager and the nodes of the region.
- The maximum uniform region. The uniform number of nodes in the region is necessary to protect against overloads or outages of individual managers in the network.

Since we use several criteria for solving this problem, you need to identify the main criteria on which the solution will be evaluated. In this algorithm, the main criterion is the distance. At first, algorithm looks for the best solutions by the distance, and then as far as possible uniform attaches nodes to the median and forms the regions. In this approach, the losses can be in the uniform number of nodes in the regions.

As input data it has:

1) Graph of the network.
2) The number of managers, (p).

As an output a designer receives the following information:

1) nodes ID to which to connect managers.
2) nodes ID, which are part of every manager's region.
3) Route of data transfer between the manager and the nodes of the region.

To describe p-median algorithm, we use a simplified example of a network with Mesh topology and size of 3x3, shown in Fig. 5. For example, take p = 2, i.e., you must place 2 manager.

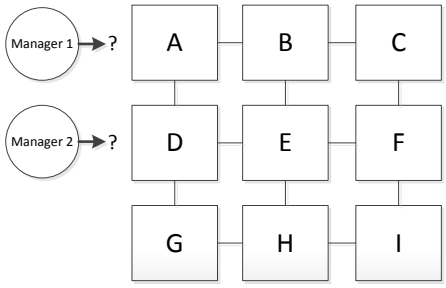


Fig. 5. An example network with Mesh topology and size of 3x3 for solving p-median problem, where p=2.

1) Create a matrix of shortest distances MSD.

	A	B	C	D	E	F	G	H	I
A	0	1	2	1	2	3	2	3	4
B	1	0	1	2	1	2	3	2	3
C	2	1	0	3	2	1	4	3	2
D	1	2	3	0	1	2	1	2	3
E	2	1	2	1	0	1	2	1	2
F	3	2	1	2	1	0	3	2	1
G	2	3	4	1	2	3	0	1	2
H	3	2	3	2	1	2	1	0	1
I	4	3	2	3	2	1	2	1	0

Fig. 6. Matrix of shortest distances MSD

2) Sort the rows by ascending distance.

The index shows the distance from median to nodes in the row (Fig. 7).

3) Choose two rows (since p=2) and remove from this rows median nodes. At this step, median nodes are H and B.

In Fig. 7 nodes in the left part of the table are potential medians, and all that is in the right side of the table - attachable nodes. We removed median nodes in the right table side of the selected rows, as we do not consider the interaction between managers. In Fig. 7 the selected rows are in green, and median nodes in the right table side are in red.

A	A_0	B_1	D_1	C_2	E_2	G_2	F_3	H_3	I_4
B	B_0	A_1	C_1	E_1	D_2	F_2	H_2	G_3	I_3
C	C_0	B_1	F_1	A_2	E_2	I_2	D_3	H_3	G_4
D	D_0	A_1	E_1	G_1	B_2	F_2	H_2	C_3	I_3
E	E_0	B_1	D_1	F_1	H_1	A_2	C_2	G_2	I_2
F	F_0	C_1	E_1	I_1	B_2	D_2	H_2	A_3	G_3
G	G_0	D_1	H_1	A_2	E_2	I_2	B_3	F_3	C_4
H	H_0	E_1	G_1	I_1	B_2	D_2	F_2	A_3	C_3
I	I_0	F_1	H_1	C_2	E_2	G_2	B_3	D_3	A_4

Fig. 7. Sorted distance matrix

4) Derive a new table AMD that contains attachable nodes, medians, to which they are attached, and the distance to them in ascending order (Fig. 8).

5) Remove attachable nodes, to which the distance is not minimal. In Fig. 8 removed nodes are in red.

6) *From the resulting table one can uniquely identify nodes that can be attached to only one median.* In Fig. 8 uniquely attachable nodes are in green, removed nodes are in red.

Attachable nodes	A	C	E	G	I	D	F	A	C	G	I
Medians	B	B	B,H	H	H	B,H	B,H	H	H	B	B
Distance	1	1	1	1	1	2	2	3	3	3	3

Fig. 8. Matrix AMD

7) Distribute remaining nodes on the medians with the maximum uniformly attachment.

Nodes E, D, F can be attached to the both medians - B and H. These three nodes cannot be uniformly divided between the two medians, so one median gets two nodes, a second median gets one node.

8) Check solution's set coverage.

In this example, the set of vertices is V = {A, B, C, D, E, F, G, H, I}, a median subset is V_P = {B, H}, the subset of attached vertices is V_A = {A, C, D, E, F, G, I}. Since $V_P \cup V_A$ = V, then the solution is correct.

The obtained solution is shown in Figure 2. Nodes A, C, D, E are in region for the manager at the node B. Manager at the node H attaches nodes F,G,I [7].

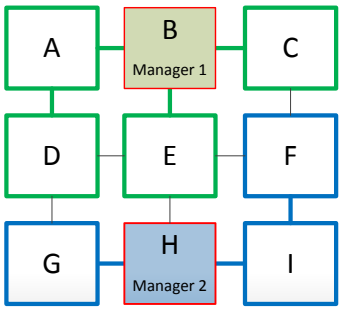


Fig. 9. Solution for p-median problem for network with Mesh topology and size of 3x3, where p=2

The P-median problem is NP-complete, and the time to solve it depends exponentially on the input data size. However, in view of the fact that currently the network has a small number of units (not more than 224 SpaceWire standard) [8], The P-median problem still can be solved using exact algorithm. In this paper, is used the exact algorithm for solving the P-median problem. We have done several tests to measure the operating time of the algorithm on different topologies with different number of regions. The Fig. 10 shows a plot of the dependence of time consumption and input data.

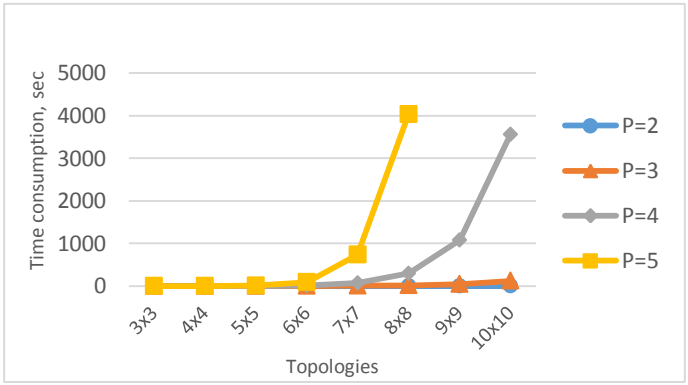


Fig. 10. Dependence of time consumption and input data for exact algorithm.

As can be seen from the graph the longest it searches a solution for the network 8x8 with five regions and network 10x10 with four regions. Solutions for these configurations have been received in about an hour.

IV. Comparison of the results, obtained by these approaches

In an experiment to compare two approaches of finding the points of connection managers in network, we have chosen networks with different topologies and different number of nodes. In the presented topologies in rectangles and squares represent switches, and circles – labeled terminal nodes; nodes highlighted with a solid color – connected managers. Nodes in the region are circled by the color of the appropriate manager.

1) *Arbitrary topology*.

Contains 61 node 22 of the switch 39 and terminal nodes). In this topology three centres of the connection managers of the PnP were searched.

Fig. 11 shows the solution obtained using the heuristic algorithm of graph partitioning into subgraphs. Regions highlighted in red (25 nodes), blue (node 29) and brown (7 nodes). The maximum distance from the Manager to the nodes in the region is 4-hop (number of switches on the path between the manager and the most remote from it node in the subgraph).

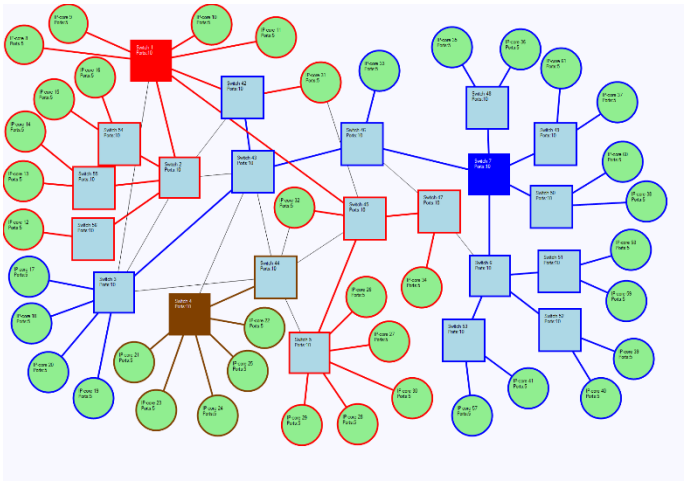


Fig. 11. An example of the worst solutions of the heuristic algorithm for an arbitrary topology

Fig. 12 shows the solution obtained by using algorithm p-median. Regions highlighted in red (20 nodes), blue (21 nodes) and brown (20 nodes). The maximum distance from the manager to a node in the region is 3 hops.

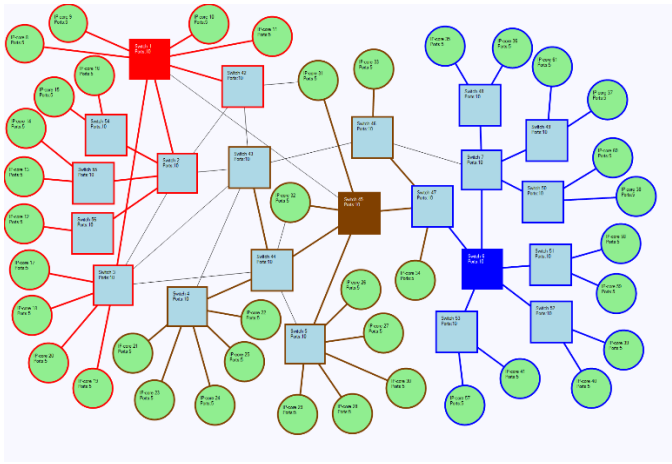


Fig. 12. Example of the solution of exact algorithm for arbitrary topology

2) Mesh 8x8.

This regular topology contains 128 nodes, 64 switch and 64 terminal node). On this topology, were searched five managers. It is important to note that to each switch is connected one terminal node. For simplicity of illustration, we do not represent terminal nodes in the network.

Fig. 13 shows the solution obtained by the heuristic algorithm of partitioning a graph into subgraphs. Regions highlighted in red (28 nodes), green (26 nodes), blue (28 nodes), pink (28 nodes), and yellow (18 nodes). The maximum distance from the manager to a node in the region is 3 hops.

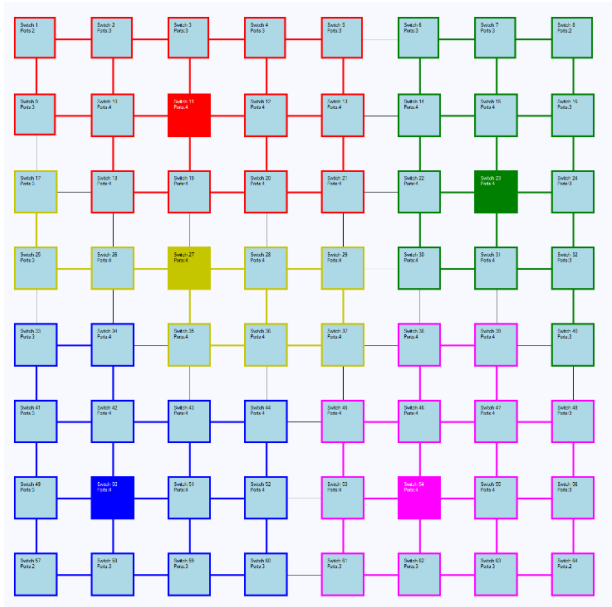


Fig. 13. Example of the best solution of a heuristic algorithm for Mesh topology with size 8x8

Fig. 14 shows the solution obtained by using algorithm p-median. Regions highlighted in red (26 nodes), green (26 nodes), blue (26 nodes), pink (26 nodes), and yellow (24 nodes).The maximum distance from the manager to a node in the region is 3 hops.

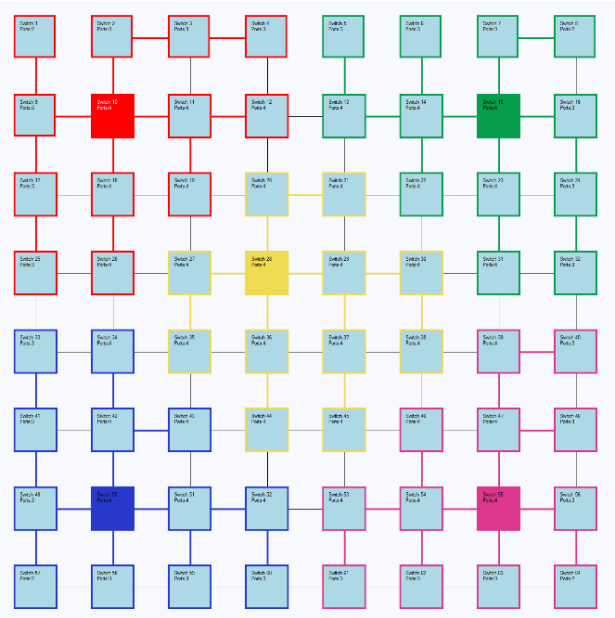


Fig. 14. Example of the solution of exact algorithm for Mesh topology with size 8x8

V. SIMULATION RESULTS

The data below were obtained using simulation software NetSim focused on SpaceWire networks simulation [4]. The network was simulated by simultaneous operation of a manager PnP with three devices in the region. The SpaceWire packets do not go beyond the region, that is, the majority of traffic was exclusively within each region and had no effect on downloading of the neighboring regions.

To perform the configuration of all devices within each region the RMAP protocol is used; the packet size is 23 bytes to 28 bytes.

A. *Mesh 8x8.*

According to the obtained results of simulation were plotted diagram for channel load for five managers. For each manager defined input (the columns 'In') and output (columns 'Out') channel download.

In the Fig. 15 highlighted in red are the simulation results of the worst, in orange are the best solutions obtained by the approach of partitioning the graph into subgraphs, in blue the results of the exact algorithm. BA-14 – a manager of the red region, BA_28 is the manager of the green region, BA_41 is the manager of the yellow region, BA_55 is the manager of the blue region, BA_69 manager of the pink region.

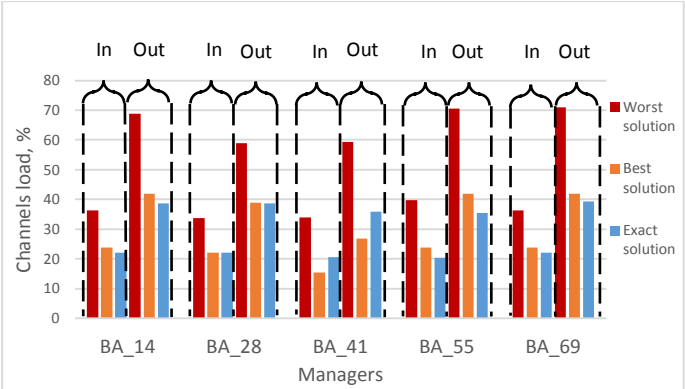


Fig. 15. Loading of channels for all managers in Mesh topology with size 8x8

From Fig. 15 it is seen that the heuristic method may produce a result about 1,5 times worse from optimum. The load on input and output ports will sharply increase. However, the best distribution results will not differ much from the optimal.

An important feature of the division into regions is the uniformity of the number of nodes in the regions. The figure shows the distribution of nodes in the regions.

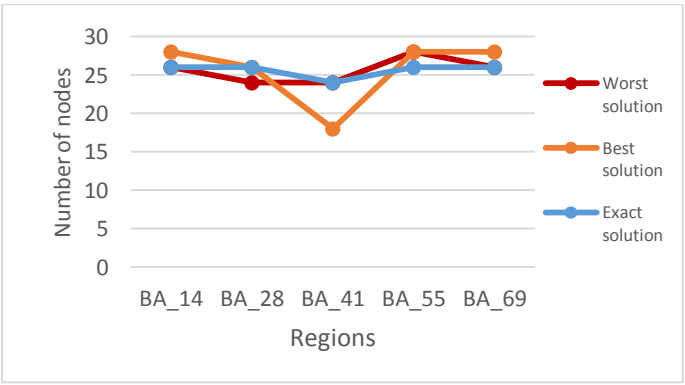


Fig. 16. The uniformity of the regions worst, best and exact solutions

From the diagram, it is possible to note, that even the best solution of a heuristic algorithm can strongly lose by criterion of uniformity of regions, while the exact algorithm always outputs a solution with evenly loaded regions.

B. *Arbitrary topology.*

The results of simulation of solutions to partition the arbitrary topology using heuristic and exact methods showed that when splitting the graph into subgraphs, it is important to consider not only the number of nodes in the region, but also their degree- the number of edges leaving the given vertex. In other words, in case of a network should consider how many active ports are involved in nodes that are distributed in the region. We did not consider this option and the figure shows simulation results in which the exact algorithm gives the load on the input and output ports bigger than the best solution of the heuristic approach.

At every switch within the region sent the total number of packets equal to the sum of the number of active ports (used in this network topology) and the number of adjustable rows of

the routing table. The routing table mentioned the same for all switches, but due to different numbers of active connections on each switch there was a difference in the traffic for each of them. For each terminal node has sent a fixed number of packets. But it could also have an impact on the loading of channels, because a different number of terminal nodes present in solutions of the heuristic and the exact algorithms.

On Fig. 17 BA-40 is a manager of the red region, BA_41 is a manager of the blue region, BA_42 is a manager of the brown region.

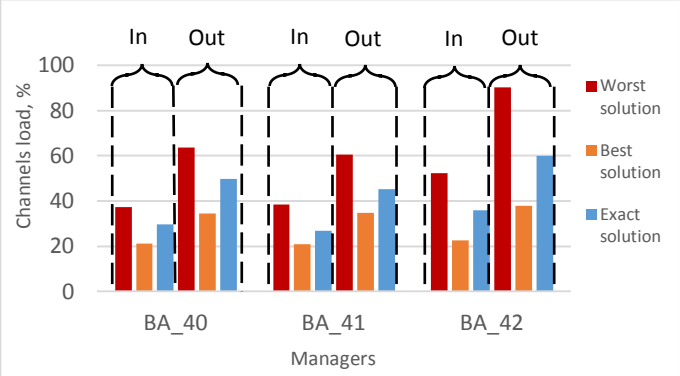


Fig. 17. Loading of channels for all managers in arbitrary topology

Also it presents a paradox due to the fact that the best solution of the heuristic algorithm, the regions formed so that each region was nodes and with big and with fewer active ports. While the algorithm based on algorithm P-median was placed in the same region nodes with large number of ports, and in another region – with less, although the number of nodes in regions of exact and heuristic algorithms is the same.

VI. Conclusion

This article presented two approaches to determine the PnP managers of locations in networks and their regions. The first approach is based on a heuristic algorithm, the second – on an algorithm that gives guaranteed exact solution. The selected topology is represented as an undirected graph, divided into regions. Then we carried out simulation with traffic shaping entirely inside each formed network region.

The results of the network partition in case of the heuristic algorithm may be 1.5 times worse than the optimal result obtained by exact method. However, the best solution heuristic algorithm is close to optimal, but to obtain it you need multiple iterative runs of the algorithm. Heuristic approach is cheaper to apply on large networks with a large number of managers, however, if there are limited number of PnP managers (up to 5), it is better to use the exact algorithm, which gives optimal solution according to the criteria of distance and load.

Acknowledgement

The research leading to these results has received financial support from the Ministry of Education and Science of the Russian Federation under grant agreement no. RFMEFI57814X0022.

References

[1] D.P. Buvailo, V.A. Tolok, Fast high-performance algorithm for irregular graphs partitioning, Zaporizhzhya state university, Ukraine № 2, 2002.

[2] E. Mineka, Optimization Algorithms for Networks and Graphs. M.: Mir, 1981.

[3] V. V. Bereznitsky, N. V. Lukyanova, “Research and development of an algorithm for finding the shortest path in the graph”. Electronic Journal Youth Science and Technology herald, 2012.

[4] A. Eganyan, L.Koblyakova, E. Suvorova, “SpaceWire Network Simulator”, Proceedings of the 3rd International SpaceWire Conference, 2010, pp.421-425.

[5] N.Cristofides, Graph Theory. An Algorithm Approach. New York: Academic Press, 1975.

[6] P. Hansen, B. Jaumard, Cluster Analysis and Mathematical Programming. Mathematical Programming 79, 1997, pp. 191-215.

[7] E. Suvorova, L. Kurbanov, N. Matveeva, “Using P-median Algorithm for 3D Network-on-Chip Design”, Proceedings of the 18th FRUCT & ISPIT Conference, 2016, pp. 553-557.

[8] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire, Links, Nodes, Routers and Networks”, Issue 2, European Cooperation for Space Data Standardization, July 2008.

A New SpaceWire Protocol for Reconfigurable Distributed On-Board Computers

SpaceWire Networks and Protocols, Long Paper

Ting Peng, Benjamin Weps, Kilian Höflinger
Simulation and Software Technology
German Aerospace Center
Lilienthalplatz 7
38108 Braunschweig, Germany
ting.peng@dlr.de, benjamin.weps@dlr.de,
kilian.hoeflinger@dlr.de

Kai Borchers
Institute of Space Systems
German Aerospace Center
Robert-Hooke-Str. 7
28359 Bremen, Germany
kai.borchers@dlr.de

Daniel Lüdtke, Andreas Gerndt
Simulation and Software Technology
German Aerospace Center
Lilienthalplatz 7
38108 Braunschweig, Germany
daniel.luedtke@dlr.de, andreas.gerndt@dlr.de

***Abstract*—There are several standardized protocols based on SpaceWire which provide data exchange between several nodes. SpaceWire is also suitable for interprocess communication (IPC), by the help of higher level protocols. However, currently there is no standardized protocol which is targeting IPC on SpaceWire networks. This paper proposes a protocol, which uses the capabilities of SpaceWire to build up networks for distributed computing on a spacecraft. The core of this protocol is the IPC mechanism for communication between the nodes and methods to support a reconfiguration of the network. A key feature of this protocol is an interface for a reconfiguration mechanism, which can be implemented on application level. This enables the utilization of unreliable commercial off the shelf (COTS) nodes, allowing system recovery from erroneous state. Additionally, the reconfiguration can be used to adapt the distributed computer to different mission phases. The protocol has the potential to build the foundation of a distributed on-board computer consisting of COTS components. Such distributed computer could be capable of fulfilling high performance demands as well as high reliability needs. Though, the protocol itself is not restricted to be used solely in fully-featured reconfigurable distributed systems. The IPC methods can be applied stand-alone as well, to establish a lightweight communication between nodes on a SpaceWire network by excluding the reconfiguration parts of the protocol.**

***Index Terms*— SpaceWire, Network, Protocol, Reconfigurable, Interprocess Communication, COTS, High Reliability.**

I. Introduction

Distributed systems with COTS components use multiple computing nodes to share the workload and offer significantly higher computing performance than currently used space-qualified on-board computers. It is necessary to offer complex IPC services and satisfy strict requirements for satellite missions, such as real time and reliable transmission as well as high transmission speed. Reliability of COTS can be realized via redundancy by the execution of equivalent tasks on different nodes and by the reconfiguration of nodes and tasks, i.e., migration of tasks to other available nodes after some nodes fail. The distributed system should support upgrade, maintenance and failure detection, isolation as well as recovery.

SpaceWire is suitable for IPC with further protocols. However, currently there is no standardized protocol that is explicitly targeting IPC on SpaceWire networks and to be utilized in reconfigurable distributed on-board computers. Therefore, a new protocol, based on SpaceWire, is necessary to support the reconfigurable distributed on-board computers.

We will introduce a new protocol called SpaceWire-IPC, which is beneficial for reliable and fault tolerant distributed on-board computers

The paper is organized as follows. Section II presents the related work, which was taken into account during development of our proposed SpaceWire-IPC protocol. Section III describes the requirements for the protocol, derived from our project. Structure and properties of the new protocol SpaceWire-IPC are introduced in section IV. Finally, section V provides the comparison between SpaceWire-IPC and already existing protocols, followed by a conclusion in section VI.

II. Related Work

This section provides an overview of existing SpaceWire compatible protocol specifications. With the exception of

SpaceWire-R, all protocols are referenced and officially adopted by an according ECSS standard [1]. Additionally, the trends of IPC and reconfiguration in space systems are presented.

A. Overview of Existing SpaceWire Protocols

The subsequent paragraphs list SpaceWire protocols, which were considered for the development of the distributed on-board system.

1) Remote Memory Access Protocol (RMAP)

The Remote Memory Access Protocol (RMAP) protocol, defined by standard [2], is commonly used in space applications for reading from and writing to memory in remote SpaceWire nodes. The protocol provides the ability to address destinations by the use of path or logical addressing over the *Target SpW Address* fields as defined in [3]. However, in case only logical addressing is required it is also possible to skip the *Target SpW Address*. The protocol can be directly integrated into a standard SpaceWire protocol by using the *Protocol Identifier*. By use of an *Instruction* byte, the following modes/message types are possible:

- Read command/ write command
- Verify / no verify of data before write
- Acknowledge / no acknowledge of write command
- Read-Modify-Write

If data shall be read or an acknowledgment is required, a set of *Reply Address* fields are available, which are usable for path and/or logical addressing. The source of the received command is stored inside the *Initiator Logical Address*. To prevent a lock-step limitation during communication two *Transaction Identifier* bytes are available, which allow the user to apply out of order transfers. To define the target location for read or write commands, a set of four *Address* fields, plus an additional *Extended Address* field is used.

Written and read data is secured by Cyclic Redundancy Checks (CRCs) for Header and Payload data independently.

2) CCSDS Packet Transfer Protocol (CCSDS PTP)

The CCSDS protocol is intended to encapsulate a user defined protocol that needs to be transferred through a SpaceWire network [4]. Similar to the RMAP, an arbitrary amount of *Target SpW Address* fields can be used for routing. Alternatively, the *Target Logical Address* is used to define the destination. The *Protocol Identifier* distinguishes between different SpaceWire packet types. The interpretation of data of the *CCSDS Packet* fields is user specific and defined inside the *User Application* field.

3) GOES-R Reliable Data Delivery Protocol (GRDDP)

The main purpose of GRDDP is to transfer data of sensors, telemetry and commands among peripheral instruments and the on-board computer [5]. The *Destination SLA* serves as a logical address, related to the targeted destination. To provide information about the source of the packet *Source SLA* is used. Four different packet types can be used by defining the *Packet Type* field:

- Application Data
- Acknowledge
- Reset Command
- Urgent Message Data

To detect packet loss or to order out-of-order packets, a *Sequence Number* is provided. The whole packet, except End-Of-Packet (EOP), is covered and checked by a CRC.

4) Serial Transfer Universal Protocol (STUP)

The STUP protocol, defined in [6], serves as a light weighted protocol with the intention to implement a more complex protocol, inside the data field. To define the source of the packet the *Source Logical Address* is used. Different kinds of data structures can be defined by the *Data* fields. The standard defines an example where the first data byte defines a kind of message type, which is used to interpret the left data bytes. Only write, read and read reply commands are offered in this example.

5) SpaceWire-R

SpaceWire-R is used for reliable data transmission within SpaceWire networks [7]. It is based on the GRDDP and the Joint Architecture Standard Reliable Data Delivery Protocol (JAS RDDP). SpaceWire-R provides features like multiplexing, message segmentation, reliable transfer, network traffic flow control (optional) and heartbeat signaling (optional) [7].

B. Trends in Space Systems

Several space projects use distributed on-board computers to meet the increasing demands of on-board processing ability. The On-Board Computer - System Architecture (OBC-SA) consists of two on-board computers, one of which is COTS from Freescale's PowerPC multicore CPU [8]. The High-Performance Reconfigurable Computing Space Processor (CSP)'s hardware structure is based on both COTS and radiation-hardened technologies. ISS SpaceCube Experiment Mini (ISEM)'s hardware has two CSP boards which are interconnected by SpaceWire and UART [9]. CSP aims to offer space image processing, distributed parallel computation and fault tolerance [9]. The Fault-Tolerant Distributed On-Board Computer (FTD-OBC) gains higher reliability and higher processing performance by multiple processing nodes connected by CAN buses of 1 Mbps [10].

III. Protocol Requirements

Inspired by the rise of distributed computing techniques and advantages of SpaceWire, the project Scalable On-board computing for Space Avionics (ScOSA) and its predecessor project On-board Computer - Next Generation (OBC-NG) at German Aerospace Center (DLR) use COTS hardware besides radiation-hardened components to establish a distributed on-board computing network, based on SpaceWire. Their goal is to leverage performance of a distributed architecture and still maintain the required reliability.

In ScOSA, three types of nodes, High-Performance Nodes (HPNs), Reliable Computing Nodes (RCNs) and Interface Nodes (IFNs) are used (see Fig. 1). HPNs are based on a Xilinx Zynq XC7Z020 architecture (CPU + FPGA) while RCNs have a LEON3 as FPGA soft-core implementation. The SpaceWire router is integrated in the FPGAs of the RCNs, the HPNs and the IFNs.

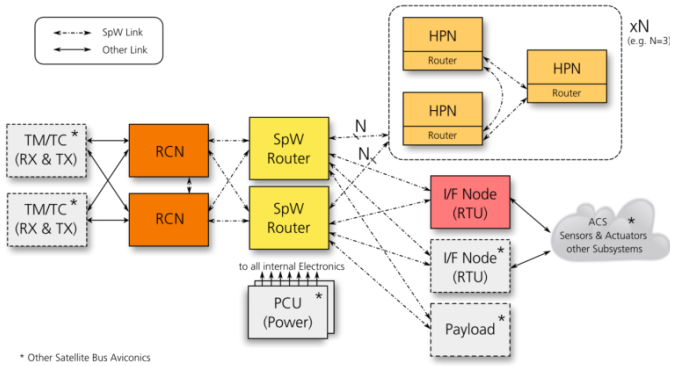


Fig. 1. ScOSA system overview

The middleware offers monitoring, task management, checkpointing and reconfiguration services for the system. These services are coordinated by three types of roles among the RCNs and HPNs: Master, Observer and Worker. A global configuration for all nodes means the deployment of the Master, Observers and Workers on RCNs and HPNs, settings of monitoring behavior, and channels availability and subscriber lists of IFNs, etc. The Master is responsible to initiate the configuration for all nodes in the network, by broadcasting the configuration command. The Master also monitors the distributed system via a periodical heartbeat mechanism and a plausibility check of some control values of application tasks. Some internal states of the application tasks are periodically sent to the mass memory storage for checkpointing. If a node fails, the Master will trigger a system reconfiguration and redistribute the tasks to other nodes. After the reconfiguration finishes, the checkpoints will be retrieved from storage back to the nodes, which are running the corresponding tasks. Two or more Observers are assigned to monitor the Master. In case the Master fails, a decision will be made to choose one Observer to take over the failed Master's tasks.

This reconfigurable distributed system intensively relies on IPC. For the consideration of scalability and throughput, a bus topology can't be used [11]. Nodes are interconnected with point-to-point links. An irregular network topology structure is used to avoid a single-point failure and to maintain flexibility [11].

To summarize the analysis of requirements for the ScOSA distributed system, the network should

- be scalable and flexible,
- be able to transfer arbitrary large messages,
- have high reliability supported by redundant routes among nodes in case of failed nodes,
- guarantee the reliability of messages delivery,
- deal with message losses,
- support monitoring, error notification and reconfiguration.

IV. Protocol Description

The ScOSA project mainly uses the SpaceWire-IPC for the communication among the distributed computing nodes. The SpaceWire-IPC is located at the transport layer of ISO-OSI model [12]. SpaceWire acts as the underlying protocol (see Fig. 2).

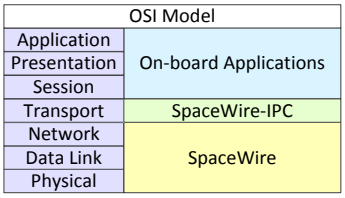


Fig. 2. Protocols and OSI model

A. Features

SpaceWire-IPC offers communication for:

- IPC among nodes
- Management services from and to Master

SpaceWire-IPC supports:

- Multiple logical nodes on one physical device
- Reliable transmissions of data as well as unreliable transmissions
- Recognition of failed connections and failover mechanisms
- Transmission of messages with arbitrary size
- Transmission of large-size messages
- Transparent use for different underlying protocols
- Multiple APIs rather than only read and write commands

The protocol is message-based, meaning that instead of streams, single messages are sent from one node to another. These messages can be reliable or unreliable.

SpaceWire offers no regulation regarding the maximum packet sizes. SpaceWire-IPC implements a sequencing technique, which allows splitting large messages into smaller packets, with their size being user-defined. The message is subdivided in packets on the sender and reassembled at its destination node. In case of reliable message, it is a bidirectional packet transfer with acknowledgments. Each packet has a checksum to verify the integrity.

B. Design Decisions

This section lists and explains the message structures used in SpaceWire-IPC (see Fig. 3).

Target SpW Address (1 byte)	.....	Target SpW Address (1 byte)
Target Logical Address (1 byte)	Potocol Identifier (1 byte)	Sender Node ID (2 bytes)
Receiver Node ID (2 bytes)	Timestamp (8 bytes)	Message Type (1 byte)
Payload Data (0 to n bytes)	Checksum (4 bytes)	EOP (1 byte)

Fig. 3. Structure of a SpaceWire-IPC Packet

1) Message Header

The message header shown in Fig. 3 is identical for all types of messages. The header contains source and destination of the packet, the timestamp it has been created and the size of the payload. While the header stays the same, the message type determines the structure of the payload data.

a) Sender Node ID and Receiver Node ID

Sender Node ID and *Receiver Node ID* are the logical addresses of the nodes that participate in this transmission. An ID determines exactly one entity in the network capable of sending and receiving messages. This does not necessarily mean, that it has to be unique for each physical node (regarding to SpaceWire Addressing), which is connected to the network. A physical node can have multiple software components running, which are able to send and receive messages.

b) Timestamp

This marks the time the packet has been created. The timestamp, together with the sender and receiver node ID, is the unique identifier for a packet.

c) Message Type

This field determines the structure of the payload data. It also indicates how the receiving node should handle this message. The possible values are listed in TABLE I.

One exception in this scheme is the *Large Message Transfer*. The bits 0-6 define the encapsulated message type as usual. The most significant bit determines if this message is part of a *Large Message Transmission.*

TABLE I. SUMMARY OF MESSAGE TYPES

Integer Value	Message Type
0	Unreliable Data Transmission
1	Reliable Data Transmission
2	Data Request
3	Data Response
4	Reconfiguration Request
5	Message Acknowledge
6	Heartbeat
7	Error Notification
128+	Large Message Transfer

d) Payload Data

To stay as versatile as possible, the payload data is just an arbitrary-sized byte array. The structure can be derived from the message type. For some message types, the array has a fixed size, other message types have a variable-sized array of data. The exact structure for each data type is described later in this paper.

e) Checksum

The checksum provides a way to check the integrity of the transmission. As it is not guaranteed that the underling protocol has a mechanism to detect erroneous messages, the integrity check will be implemented in this protocol. The protocol does not dictate a specific algorithm for the checksum. The only restriction is, that it must not exceed the size of the 32-bit value provided by this field. The value of this checksum should consider all of the previous fields of the transmission to guarantee the integrity of the whole packet.

The checksum algorithm can be selected by considering the mission requirements and available resources of the nodes in the network. An example for checksums is the CRC32 algorithm as described in the IEEE802.3 (Ethernet) Standard.

2) Data Transmissions

The *Data Transmission* types are the central message type for transmitting data inside the distributed system. The data is handled by the protocol as an arbitrary byte array. Hence, it has no influence on the handling of the message. The structure and handling of the data is not part of the protocol and has to be conducted at the application level.

The payload structure of *Data Transmissions* is the same for both, reliable and unreliable transmissions and its layout is shown in Fig. 4. The Data Size contains the number of elements of the following byte array. The data byte array contains the actual data.

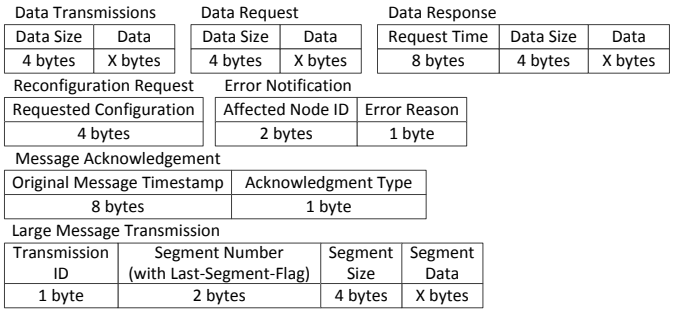


Fig. 4. Summary of payload structures

Data can be transmitted in a reliable or in an unreliable manner. Therefore, two message types are available for this purpose: reliable and unreliable *Data Transmission*. The only difference between these two types is that the receiver, when received successfully, will acknowledge the reliable *Data Transmission*. Though, the sender can resend a packet if it was lost or falsely transmitted. Unreliable messages will be dropped when received erroneous.

3) Data Request

For transmitting data with the request-response method, a *Data Request* message can be sent. This message triggers the receiver to execute an action and send data to the sender of this request.

To assign the response to the according request, the responding data will be transmitted with a *Data Response* message instead of a normal *Data Transmission*. The *Data Response* message, which is following the request, will be sent asynchronously. In that way, the action, which has been requested, can take longer time than the normal acknowledgment.

The payload of a *Data Request* has the same structure as in Fig. 4. The *Data Size* contains the number of elements of the

following *Data Byte Array*. This array contains the data that will be sent to the remote node. The structure of this array is not specified in this protocol. The data has to be parsed at the application layer.

To ensure the request will trigger an action and a data response, *Data Requests* are reliable messages, which have to be acknowledged.

4) Data Response

The *Data Response* is used to send data back to a node that has sent a *Data Request* message. This message will be sent asynchronously after the action that was triggered by the request has been executed.

The payload of a *Data Response* has also the structure shown in Fig. 4. The Request Time field contains the timestamp of the original *Data Request* that triggered this response. The Data Size contains the number of elements of the following byte array. The Data byte array contains the data that will be sent to the remote node. The structure of this array is not specified in this protocol.

To ensure the request will trigger an action and a data response, *Data Responses* are reliable messages that will be acknowledged.

5) Reconfiguration Request

The *Reconfiguration Request* notifies all nodes in the network to switch to a certain configuration. Only the Master node is capable of sending these requests, as it is the only instance authorized to define the global state of the system. Every other node than the Master node shall only be able to receive this message, but not sending it.

Reconfiguration Requests are always transmitted reliable. Nodes that are not responding to a *Reconfiguration Request* have to be disabled by a new reconfiguration.

6) Message Acknowledgment

The *Message Acknowledgment* is the central element in the reliability mechanism of this protocol. Reliably sent messages will be acknowledged with this message type. The acknowledgment can be either a positive acknowledgment, notifying that the message has been received successfully, or a negative acknowledgment to inform the sender of the original message, that it arrived erroneous.

Acknowledgments are not transmitted reliable. When an acknowledgment packet is lost, the original message will be sent again.

7) Heartbeat

The *Heartbeat* is a message type, used to check if a certain node is responsive. The *Heartbeat* itself is a request for an acknowledgment message. This mechanism allows a verification of the bidirectional communication link. The management instance sends out these *Heartbeat* messages periodically.

8) Error Notification

The *Error Notification* is used to inform the Master of the distributed system about an error that occurred. As soon as a reliable connection is not acknowledged, after a certain amount of attempts, this notification will be sent to the Master node. This message can also be used to notify other nodes that an error has occurred.

Error Notifications will be sent reliable. This enables the possibility to detect whether the sending node has lost the connection to the network, or the erroneous node is the source of the failure. If this error message cannot be delivered to a management node it can been assumed that the node itself has lost its connection to the rest of the distributed system.

9) Large Message Transmission Packet

SpaceWire itself does not limit the packet size. But to avoid long blocking of paths in the network, a restriction on packet size is defined in this protocol. Here, the *Large Message Transfer* mechanism offers a way to split a message into smaller packets, which then can be sent sequentially to the destination node. The receiver collects all parts of the messages and assembles them to the original message.

The capability to send and receive *Large Message Transmissions* is optional, if a node sends a *Large Message Transmission* to a node not capable of this feature the receiver shall reply with an *Error Notification Message*.

The *Large Message Transmission* is a special message inside the protocol. To keep the transmission size as small as possible, the complete header of the original message provided by the sending application will be integrated into the header of the *Large Message Transfer* with the modifications that the most significant bit of the message type field is set and the timestamp of each packet is independent from the timestamp of the original message.

Large Message Transmissions are always reliable transmissions. Every segment of this transmission will be acknowledged individually (either positively or negatively) according to the timestamp of the segment.

C. Behavior Description

1) Reliability

Central paradigms of the protocol are reliable messages and error detection and handling. The protocol provides guaranteed delivery services and timeout mechanism for reliable message transmission. The messages transmitted follow the reliability mechanisms that are described as follows.

a) Single Packet Messages

As every message is sent independently, the reliability mechanisms are also applied to every single message. Therefore, each message that is received and has a reliable message type has to be acknowledged. The reliability mechanism is divided into three phases: acknowledgment, resending and error notification phase.

Acknowledgment Phase:

Three different cases for a sent message have to be considered:

When the message was received successfully, a positive acknowledgment will be returned to the sender and the transmission is complete.

The second case is that the message has been received erroneous. With help of the checksum appended to every message, the receiver can check the integrity of the message. If it was received with errors, a negative acknowledgment will be sent, which triggers the sender node to switch to the resend phase.

The last possibility is the loss of the message on its way through the network. In this case, the receiver will not send any acknowledgment either positive or negative. Therefore, the sender waits a defined time for the acknowledgment to arrive and if this time passes, it will switch to the resending phase.

Resending Phase:

When a message was not transmitted successfully on the first try the sender will attempt to resend it. The number of attempts is configurable. The sender again expects a message acknowledgment. If, at a certain try, the message will be acknowledged the transmission is completed. If the limit of resending is reached the system will assume that the link to the receiver is faulty and switches to the error notification phase.

Error Notification Phase:

In this phase of a transmission, it is very likely that the receiving node has lost the connection to the network, as it does not respond, although the sender has repeatedly tried to communicate with it. Another error could be that the sender itself has lost the connection to the network and could not communicate to other nodes.

To check which node lost the connection and to inform the Master in the network about the error an error notification message will be sent to the Master node.

This error notification message is also a reliable message but it is handled differently. This message will only follow the process up to the resending phase. If that phase fails, most likely the sending node has lost its connection to the network and cannot even reach the Master node. At this point, the node should shut itself down, to save energy and not to interfere with the rest of the system.

b) Large Messages

Messages which are too large to fit into one packet should be treated as *Large Message*. For single messages transmitting segments of the *Large Message Transmission*, the reliability mechanisms work as they do for normal single messages. Additional to the reliability mechanisms for single segments, there are some extensions for the *Large Message Transmission*. When receiving the last segment of such transmission, the application has to check that no segment is missing. A *Large Message Transmission* is only successful when all segments of this transmission have been received. If segments are missing, a negative acknowledgment is sent to initiate retransmission.

2) Push Transmissions

Push Transmission are following the publish/subscribe pattern. A producer of data can have many consumers, which subscribe to it. Whenever new data is available, the producer will send the new data to all of its consumers.

The central points of this transmission are the two *Data Transmission* messages. Whenever a producer of information has new datasets, it will create a *Data Transmission* message for each subscribed node and send it.

Depending on the requirements to the delivery of the data, the application can send either a reliable or unreliable data messages.

3) Pull Transmissions

The *Pull Transmission* follows a request-response behavior. It can be used to either trigger an action on a remote node of the network or requesting specific data from it.

To start a request the initiating node has to send a *Data Request* message to the destination node. The request will always be acknowledged, which tells the requesting node that the request will be handled.

The response to these requests will be transmitted with a *Data Response* message. Additional to the normal *Data Transmission*, which is used by the push transmission, it carries the timestamp of the requesting message with it to assign the response to its requesting message.

Data Requests will be sent asynchronously to enable long responding times for the requested action and data.

4) Reconfigurations

The ScOSA system is designed to have one global configuration for all nodes. Therefore, the protocol has to provide means to distribute reconfiguration information to all nodes, to maintain a concise system state.

Reconfiguration can have several reasons. One reason is the change to a new mission phase of the system so that the nodes of the network can be assigned different tasks. Another reason can be the failing of a node so that another node has to take over the tasks of the failed node. Despite the reason, all changes of the configuration have to be initiated by the Master by sending a *Reconfiguration Request* Message. The other nodes in the network are not allowed to send this request message.

On reception of this message the receiving node will change into the so-called “reconfiguration state”. When it reaches this state, it will send all pending messages but does not accept sending new messages. Messages received in this state will be handled as usual. With this method, it can be assured that most of the messages will not get lost during reconfiguration.

After a certain timeout, which has to be configured mission-specifically, the node will delete all of its pending messages, switch into the new state and go back into running state.

The reconfiguration only affects the endpoint nodes in the network. For other network components (e.g. routers and switches), a proper protocol for configuring those components has to be chosen. In a SpaceWire network one can choose the RMAP Protocol [2] to configure the Routing tables. Therefore, the SpaceWire-IPC is implemented in that way that it does not interfere with other protocols for reconfiguring other network components (e.g. using different protocol identifiers at the underlying protocol).

5) Large Message Transmission

The *Large Message Transmission* is a special mode for transmitting messages in the distributed system. This mode of transmission provides a way to send encapsulated messages that would otherwise exceed this size restriction. Every other message used in this system can be encapsulated into a *Large Message Transmission*.

Sending of an oversized message is completely transparent to the application whatever transmission (normal or large

message transmission) is needed. The protocol implementation automatically determines if it is needed to send the message as *Large Message Transmission* depending on its size.

The size of one single packet must be defined between all nodes in the network uniformly. The data itself will be handled as an array of bytes.

The sender first assigns a unique transmission ID to this data and then separates the array into segments. These segments will then transmitted with the same Transaction ID and the corresponding sequence number. The sequence number is used to calculate the offset of this segment in the array.

On the other end of the connection, the receiver will provide a special handling of incoming *Large Message Transmissions*. Instead of notifying the application for every received packet, the handler will collect all the parts belonging to this *Large Message Transmission* according to the same transmission ID and the same Sender ID.

After receiving all parts of a transmission, the handler will reconstruct the encapsulated message and then send it to the normal handler where the original message will be handled transparently.

D. Integration with SpaceWire

The SpaceWire Specification allows custom protocols to be transported as payload. Therefore, a field in the header is reserved to specify the used protocol [1].

The SpaceWire Protocol supports two addressing modes, logical addressing and path addressing [3]. Both methods are possible with SpaceWire-IPC, but for simplicity, only logical addressing is supported by now.

The *Node ID* will be mapped to a SpaceWire logical address with the following pattern. The least significant byte will be directly mapped to the *SpaceWire Address*. The most significant byte will then determine the service running on this node. This mapping limits the maximum addressable services to 256 services per physical node and 256 physical nodes connected to the SpaceWire Network.

V. Protocols Assessment for Reconfigurable Distributed Environment

In this section, SpaceWire-IPC and other SpaceWire based protocols mentioned in Section II are assessed focusing on IPC in distributed on-board computers.

Although reliable communication in RMAP can be established by requesting acknowledgments, the protocol does not fit completely into the requirements for our distributed system. In detail the lack of distributing timestamps and especially heartbeats is a problem. Additionally, fragmentation of large data is not supported by RMAP. Besides this, a specific reconfiguration message type is required to modify the state of the distributed system.

The CCSDS PTP only serves as a frame for more complex protocols without providing properties like data validity checks or reliable data transfers, which are required for our distributed system.

Packet types of GRDDP are defined. However they are insufficient to cover all requirements given by ScOSA, such as the lack of error notification or reconfiguration handling.

For STUP, data retransmission, segmentation of large messages and flow control need to be implemented explicitly by application users. Therefore this protocol does not cover any of our requirements related to IPC communication.

Although SpaceWire-R supports reliable data transmission and heartbeat, it does not include any message types for error notification and reconfiguration. The pull request is not implemented within this protocol. SpaceWire-R can only send reliable data and lacks the unreliable data transmission. This is necessary for high-frequency transmissions, where new data will arrive quickly, and losing some packets is considered uncritical. Although, it shares some concept with the SpaceWire-IPC protocol, it is still not fully suitable for the ScOSA use case.

TABLE II summarizes these SpaceWire based protocols and SpaceWire-IPC in terms of features of IPC. As it can be seen from TABLE II, RMAP, CCSDS PTP, GRDDP, STUP and SpaceWire-R are not targeting IPC services in SpaceWire networks. However, the IPC services are necessary for a pure COTS or hybrid reconfigurable distributed on-board computers. SpaceWire-IPC offers features for IPC, supporting monitoring, management and reconfiguration, which then can be implemented on higher level.

TABLE II. Comparison of SpaceWire Based Protocols

Features	**RMAP**	**CCSDS PTP**	**GRDDP**	**STUP**	**SpaceWire-R**	**SpaceWire-IPC**
Data Correctness Check	×		×	×	×	×
Data Retransmission			×		×	×
Multiplexing			×		×	×
Segmentation / Large Message Transmission					×	×
Flow Control					×	
Keep Alive / Heartbeat / Monitoring Support					×	×
Reconfiguration Support						×
Error Notification to Manager						×
Publish /Subscribe						×
Request-Response	×			×		×

With SpaceWire-IPC, *Data Request*, *Data Response* or *Data Transmissions* can be used for application data exchange and to request or to publish state values for plausibility checks. Applications can set the timestamp for data transmission and let SpaceWire-IPC take care of the sending timestamp. Heartbeats can be used by the Master to monitor the whole distributed network and by Observers to monitor the Master or Observers of higher priorities. Message Acknowledgment is

for reliable data transmission and detecting failures of a link or no response of a node. Reconfiguration Request can be used for initial configuration, reconfiguration due to failures and reconfiguration for new-phase missions. Error Notification is to inform Master the error reason for FDIR. Large Message Transmission can meet the increasing demands of image processing on-board for earth observation activities by transferring raw large images to several nodes for parallel processing.

VI. Conclusions

In this paper we presented the SpaceWire-IPC for reconfigurable distributed on-board computers. With this protocol, SpaceWire networks can support IPC for distributed computing on a spacecraft. We highlighted the reconfiguration feature supported by the SpaceWire-IPC, which enables COTS hardware to be used on-board with reliability and fault tolerance. With COTS nodes, high performance demands can be enhanced for future applications.

Because the SpaceWire network is not fully integrated yet, it will be part of the ScOSA project to address this issue and to embed the introduced SpaceWire-IPC. Besides the physical implementation of a SpaceWire network and the proposed IPC protocol, it is also required to provide software driver support for all peripheries depending on the selected operating system.

After implementation, the measurement and performance analysis will be carried out.

References

[1] "Space engineering. SpaceWire protocol identification," ECSS-E-ST-50-51C, ESA-ESTEC Requirements & Standards Division, Noordwijk, 2010.

[2] "Space engineering. SpaceWire - Remote memory access protocol," ECSS-E-ST-50-52C, ESA-ESTEC Requirements & Standards Division, Noordwijk, 2010.

[3] "Space engineering. SpaceWire - Links, nodes, routers and networks," ECSS-E-ST-50-12C, ESA-ESTEC Requirements & Standards Division, Noordwijk, 2008.

[4] "Space engineering. SpaceWire - CCSDS packet transfer protocol," ECSS-E-ST-50-53C, ESA-ESTEC Requirements & Standards Division, Noordwijk, 2010.

[5] "GOES-R Reliable Data Delivery Protocol (GRDDP)," 417-R-RTP-0050, NASA Goddard Space Flight Center GOES-R Project, Greenbelt, 2008.

[6] P. Rastetter, U. Liebstückel and S. Fischer, "STUP SpaceWire Protocol," SMCS-ASTD-PS-001, 2009.

[7] "SpaceWire-R," SCDHA 151-0.4, Japan Aerospace Exploration Agency (JAXA), Institute of Space and Astronautical Science (ISAS), 2015.

[8] "Project information OBC-SA," [Online]. Available: https://scrivito-public-cdn.s3-eu-west-1.amazonaws.com/fokus/public/57b85e4561eb7de5/1683f2d1b44c512887644ed0eac105fd/Projektblatt_OBCSA_EN.pdf. [Accessed 15 August 2016].

[9] C. Wilson, J. Stewart, P. Gauvin, J. MacKinnon, J. Coole, J. Urriste, A. George, G. Crum, E. Timmons, J. Beck, T. Flatley, M. Wirthlin, A. Wison and A. Stoddard, "CSP Hybrid Space Computing for STP-H5/ISEM on ISS," in *Small Satellite Conference*, Logan, 2015.

[10] M. Fayyaz and T. Vladimirova, "Fault-Tolerant Distributed approach to satellite On-Board Computer design," in *2014 IEEE Aerospace Conference*, Big Sky, 2014.

[11] D. Lüdtke, K. Westerdorff, K. Stohlmann, A. Börner, O. Maibaum, T. Peng, B. Weps, G. Fey and A. Gerndt, "OBC-NG: towards a reconfigurable on-board computing architecture for spacecraft," in *Proceedings of IEEE Aerospace Conference*, Big Sky, Montana, 2014.

[12] I. T. Union, "X.200: Information technology - Open Systems Interconnection - Basic Reference Model: The basic model," 11 June 1994. [Online]. Available: http://www.itu.int/rec/T-REC-X.200-199407-I. [Accessed 08 August 2016].

Poster Presentations

Design and Analysis of SpaceWire Hot Backup Redundant Network

Session Networks and protocols, Short Paper

XiaoSu YI
School of Instrumentation Science and Opto-electrionics Engineering
Beijing University of Aeronautics and Astronautics
Beijing, China
yixiaosu@buaa.edu.cn

CongLing TAO，HuaSong ZENG
School of Instrumentation Science and Opto-electrionics Engineering
Beijing University of Aeronautics and Astronautics
Beijing, China
taocongling@163.com

***Abstract*—SpaceWire is a new bussing technology provided by European Space Agency. SpaceWire enhances the functions of error detection and error recovery [1]. Considering the application background, the reliability of SpaceWire network must be guaranteed. The paper develops a new hot back-up method for improving the reliability of SpaceWire network. The method uses the one of the reserved bits of time-code in SpaceWire to shift working mode of the SpaceWire router to realize the hot backup function and enhance the reliability of the SpaceWire network.**

***Index Terms*—SpaceWire, SpaceWire router; reserved bits of time-code；reliability; hot backup;**

I. INTRODUCTION

SpaceWire (referred as SpW) bussing technique has high speed (2MB/s-400MB/s), full-duplex and point-to-point serial data communication link and well EMC characteristics. It was applied to the project of Rosetta spacecraft and the project of Mars fast train. SpW router is an indispensable part of SpW network, every SpW node can communicate with another node through router [2]. In a SpW network, SpW time-codes provide a means of distributing time information across a SpW system. Time can be distributed across a large network with relatively low jitter. According to SpW standard, not all time-codes are used, there are two bits reserved for the future applications if the standard needs further development.

Due to the special application background of aerospace, the requirement for reliability of the link communication is very high. Hot backup redundant network is a direct and effective way [3] [4] [5] to guarantee reliability of network. In this paper, a method of hot backup function of SpW network redundancy with SpW router IP core is proposed by studying SpW standard and protocols. Based on one reserved time bit, the method can switch the SpW router working mode in the case of link failure. By doing so, the network can still ensure the normal communication function when the link failure occurs.

II SPACEWIRE OPERATION MECHANISM

A. Time-Code

In the SpW standard, the time-code is defined and the transmission priority of time-code is the highest, which is used to guarantee the whole network time synchronization. The time code is used to distribute system time over a SpW network, which comprises ESC followed by a single data character holding six bits of the system and two reserved bits [6], as shown in Figure 1:

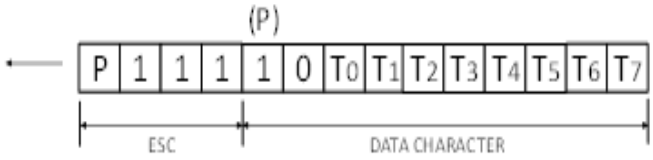


Fig 1 time-code structure

***B.** The relation between time-code and time-slot*

The time interval between the two valid time codes in the SpW standard is defined as a time-slot.

The relation between time-code and time-slot is shown in Figure 2[7]:

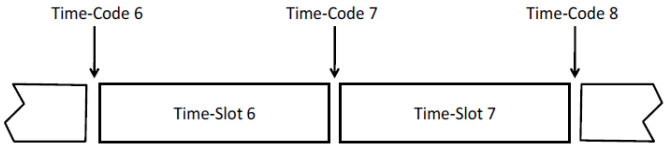


Fig 2 the relation between time-code and time-slot

When the network is running, the main control node sends the time-code across the SpW network. SpW router receives an effective time-code, updates the internal time-code counter and sends to all the nodes connected to it. The node receives the time-code sent by the SpW router, which means the beginning of a time-slot. If the SpW schedule is assigned

to a node time-slot, and the value of the time-slot is the same as the value of the time-code, the device can send data [8].

III RESEARCH AND IMPLEMENTATION OF REDUNDANT HOT BACKUP

Due to the strict load requirements, the space mission is more suitable for the high utilization of hardware resources of the hot backup.

In the paper, the SpW network hot backup method is used to realize the redundancy, which makes the reliability of SpW network improved. The structure of the redundant scheme is shown in Figure 3[9].

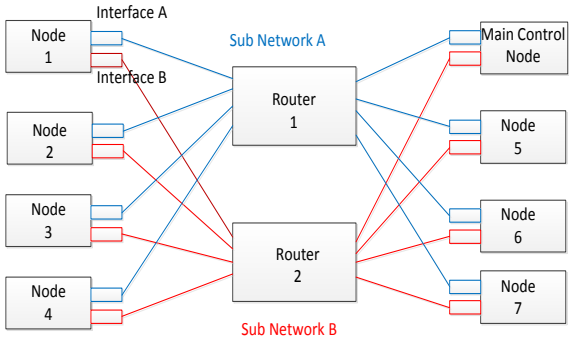


Fig 3 SpW redundancy hot backup network

In Figure 3, SpW network contains two routers that are router 1 and router 2. Each node in the network has two SpW interfaces, A and B. Through the link between interface A and router 1, a sub network A is formed. Likewise, a sub network B is formed. When the network is operated, the two interfaces of each node and the two routers all are turned on the full function.

The SpW router has link status register and time-code enable register [10]. The link status register can record and store a real-time SpW link state. Time-code enable register in the router controls where the received time-code goes, so that the router can forward the specified time-code to the appointed node. At work, the SpW link state can be read by the router link status register. For example, when a port link of router is disconnected, the information will be recorded by the port link status register in real-time.

A. Method for realizing SpaceWire network redundancy hot backup

Based on the research of SpW router and SpW standard, a platform of System on Chip (referred as SoC) system is developed in order to realize the SpW network redundancy hot backup. The structure of SoC system is shown in Figure 4, including the AXI-Lite bus, MicroBlaze processor and chip RAM, etc. The SpW router link register and time code enable register are connected with the SoC system through the SpW interface controller. When they are connected to the SoC system, they can pass data information into the MicroBlaze processor through the AXI-Lite bus. By reading the link state recorded on the link register, the processor judges the link failure in real time. When the link failure is judged, the processor will find a way to solve the problem based on the pre-designed algorithm to eliminate the fault, thus to ensure the normal operation of the SpW network system.

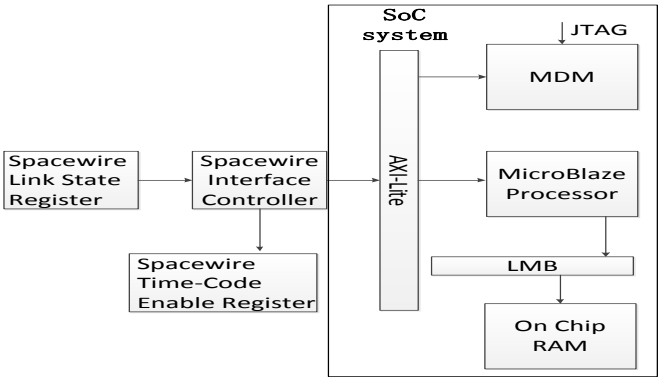


Fig 4 the structure of SoC system

B. SpaceWire network redundancy hot backup operation mode

In SpW standard, SpW time-code reserved bits T_6 and T_7 are not used. The proposed method uses one of the reserved bits which could be T_6 or T_7. For discussion convenience, T_6 is used.

The operation mechanism is described as follows:

The nodes on the SpW network are divided into two groups. Every node in each group only communicates with the node in the same group. As shown in Figure 3: node 1, node 2, node 3, node 4 are assigned to the sub network A, which receives the T_6 for 1 of the time-code as valid; node 5, node 6, node 7 are assigned to the sub network B, which receives the T_6 for 0 of the time-code as valid; There is no communication between the sub network B and the sub network A.

When network is operated, the main control node broadcasts time-code to all nodes on the network through both A and B interface. When there comes a new time-slot, each interface sends a couple of time-code, which has same system time value (T_0~ T_5), but T_6 are 1 and 0 respectively. While, router 1 forwards the time-code which T_6 is 1, driving sub network A; router 2 forwards the time-code which T_6 is 0, driving sub network B. The nodes in sub network A exchange data through router 1, and the nodes in sub network B exchange data through router 2. So far, network identification and the partition function can be achieved through the difference of time-code reserved bit T_6. Two routers drive sub network A and B according to the mode setting, two sub networks work independently according to the pre-designed schedule.

C. Failure diagnosis and solution

To illustrate how the link failure is diagnosed and solved, the paper discusses three typical link failure cases.

Case A: one or multiple effective link failures in the same sub-network occur, as shown in figure 5. The effective link here refers to a link with the same label as the network identified in the normal operation, such as the link A in the network A.

Case B: one of routers is broken or all link failures existing in the same sub network, or the link between the main control node and the router is broken, as shown in figure 6.

Case C: there are both the effective link failures and the redundant link failures, but the failure doesn't occur in the two links which are connected with the same node, as shown in figure 7.

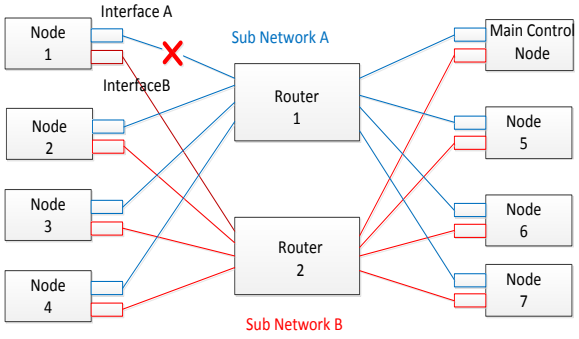


Fig 5 link failure-case A

For case A, assuming the link between the node 1 and the router 1 is broken. When such fault happens, the connection of node 1 with router 1 through interface A is broken, but the link between node 1 and router 2 through interface B is still normal. Under such a condition, to keeping the data transmission operates normally, the operation mode of each router is switched.

The detailed operation mechanism is described as follows: the SoC system judge the fault through the link state register that is in the router 1, then it will change time code enable register, making router 1 send the time code in which T_6 is 0, and at the same time, router 2 send the time code which T_6 is 1. By doing so, the operation mode of router is switched and the data flow can bypass the fault link, which will not affect the two sub networks.

For case B, assuming the router 1 is broken.

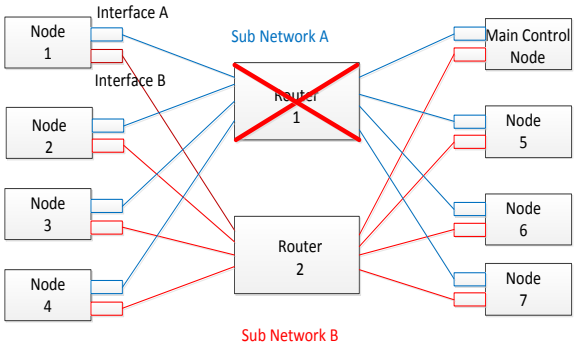


Fig 6 link failure-case B

When the fault status B appears, router 1 fault causes sub network A to be paralyzed, so any sub network could not continue to transmit data through router 1. To keep data transmission, router 2 must take the work of the router 1. Two sub networks integrate as one network. Under such a case, SoC system could not read the link state through the router 1, thus judge router 1 has something wrong. SoC system will change Router 2 operation mode, enabling router 2 can forward T_6 for 0 of the time-code and T_6 for 1 of the time-code. All nodes are in the sub network B and data exchange is carried out through the router 2. In this way, there are two sets of scheduling tables at the same time in sub network B. If two or more nodes need to send data in the same time slot, competition for the link bandwidth is required. This will result in router load increment, reducing the efficiency of the entire network.

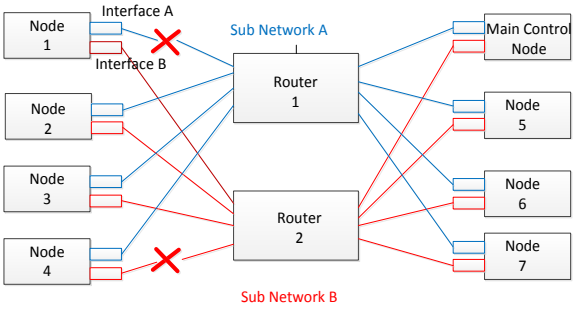


Fig 7 link failure-case C

For case C, assuming that link A between node 1 and router 1 is broken, and link B between node 4 and router 2 is broken. When such a fault occurs, the solutions for case A and case B could not be used to restore the data transmission between node1 and node 4. The method against this kind of fault is that SoC system reads link state through the related link status register of router 1 and router 2, diagnoses the fault, enables both router 1 and router 2 to forward time code with T_6 is 1 or 0. With this approach, nodes in sub network A can again receive the time code with T_6 is 1, and so do nodes in sub network B again receive the time code with T_6 is 0.

IV SPACEWIRE ROUTER TRANSMIT TIME-CODE SIMULATION

In order to verify if SpW router can transmit corresponding time-code according to link state register and time-code enable register state, some simulations were done.

Considering the situation of two nodes connected with a router, a simulation is carried out. Where node a as the main control node that is responsible for sending the time-code to SpW router, node b as the slave node that is responsible for receiving the time-code, transmitted by SpW router.

The simulation results are shown in figure 8, figure 9 and figure 10.

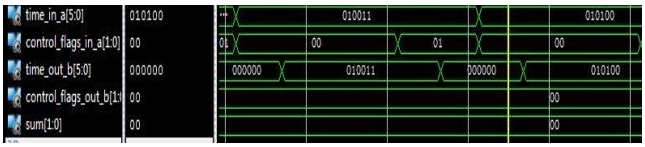


Fig 8 Simulation result for transmitting T_6 is 0

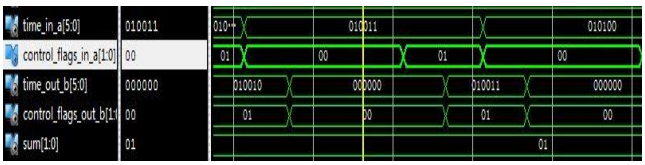


Fig 9 Simulation result for transmitting T_6 is 1

Fig 10 Simulation result for transmitting T_6 is 0 or 1

In figure 8, figure 9 and figure 10, **control_flag_in_a** are the reserved bits T_7, T_6, sending by the main control node. t**ime_in_a** is the time-code effective value T_0-T_5 sending by the main control node. **time_in_b** is the time-code effective value T_0-T_5 received by the slave node b through SpW router. **control_flags_in_b** is the reserved bits T_7, T_6 received by the slave node b. **sum** is SpW router time-code enable register. When **sum** is '00', T_6 is 0 can be transmitted through the SpW router, T_6 is 1 could not be transmitted through the SpW router. When **sum** is '01', T_6 is 1 can be transmitted through the SpW router, T_6 is 0 could not be transmitted through the SpW router. When **sum** is '10', no matter T_6 is 0 or 1 can be transmitted through the SpW router.

As shown in figure 8, **sum** is always '00', time-code can only be transmitted when **control_flags_in_a** is '00'. When **control_flags_in_a** is '01', time-code did not be transmitted.

As shown in figure 9, **sum** is always '01', time-code can only be transmitted when **control_flags_in_a** is '01'.When **control_flags_in_a** is '00', time-code did not be transmitted.

As shown in figure 10, **sum** is always '10', time-code can be transmitted when **control_flags_in_a** is '00' or '01'.

From this, it is proved that the method in this paper is feasible.

V RELIABILITY ANALYSIS

The combination of SpW redundancy hot backup network concept and the idea of application of a reserved time-code bit improve reliability of SpW network system. To prove this, some reliability comparisons are made among three operation modes: a SpW network without a hot backup, a SpW network with a cold backup, a SpW network with a hot back up. The result is shown in table 1.

In order to facilitate the reliability analysis, it is assumed there is only the link failure in a SpW network. Suppose that there are 6 nodes in this network. Suppose a single link failure rate is P in the system, and the probability of all link having failure is 0.

TABLE 1 RELIABILITY ANALYSIS

	The number of Links	One Link Failure	Two Link Failure	Three Link Failure
No Backup Network	6	Work un-properly	Work un-properly	work un-properly
Cold Backup Network	12	Still work properly	Check Note 1	Check Note 3
Hot Backup Network	12	Still work properly	Check Note 2	Check Note 4

Note 1: there are two cases of link failure in cold backup network. (1) Failures occur on the working link or backup link. (2) Failures occur on the working link and backup link respectively. For the former, the system is not affected .For the latter, the system does not work properly. The network failure rate is calculated based on the above assumptions is

$$P_1=\frac{(C_6^1+C_6^2\times C_2^1)}{C_{12}^2}\times P^2\times (1-P)^{10}=\frac{6}{11}\times P^2\times (1-P)^{10} \quad (1)$$

Note 2: there are two cases of link failure in hot backup network. (1) Failures occur on the working link or backup link. (2) Failures occur on the working link and backup link respectively. For the former, the system is not affected. The network can work normally and can work under the dual network mode without damaging the performance. For the latter, two situations might happen: 1. Faults occur on different links of different devices. The network can work normally, but with sacrifice of the dual network performance. 2. Fault occurs on the two links of the same nodes. The network could not work properly. The network failure rate is calculated based on the above assumptions is

$$P_2=\frac{C_6^1}{C_{12}^2}\times P^2\times (1-P)^{10}=\frac{1}{11}\times P^2\times (1-P)^{10} \quad (2)$$

Comparing the formula (1) and formula (2), a conclusion can be drawn that reliability of a SpW network with a hot back up is 6 times higher than the reliability of a SpW network with a cold backup. Figure 11 also shows the comparison result of reliability of the above operation modes.

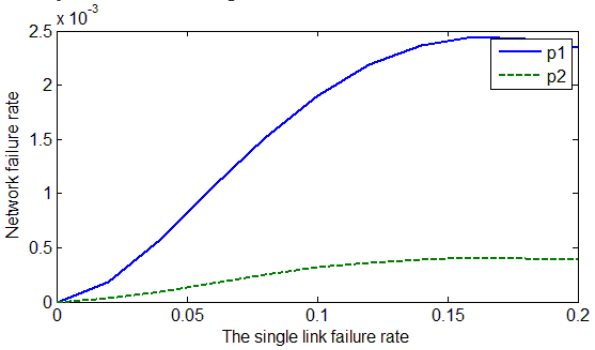


Fig 11 the comparison between P_1, P_2

Note 3: There are three link failures but analysis is as same as note 1's. When a fault occurs on a working link or a backup link, the network can work properly. Otherwise, it will not work properly. The network failure rate is calculated based on the above assumptions is

$$P_3=\frac{(C_{12}^3-C_6^3\times 2)}{C_{12}^3}\times P^3\times (1-P)^{9}=\frac{9}{11}\times P^3\times (1-P)^{9} \quad (3)$$

Note 4: Similar to the note 2. When the fault occurs on the two links of the same device, the network could not work properly. Otherwise, they are able to work properly. The network failure rate is calculated based on the above assumptions is

$$P_4=\frac{(C_6^1\times C_{10}^1)}{C_{12}^3}\times P^3\times (1-P)^{9}=\frac{3}{11}\times P^3\times (1-P)^{9} \quad (4)$$

Comparing the formula (3) and formula (4), it is obvious that reliability of a SpW network with a hot back up is 3 times higher than the reliability of a SpW network with a cold backup. Figure 12 also shows the comparison result of reliability of the above operation modes.

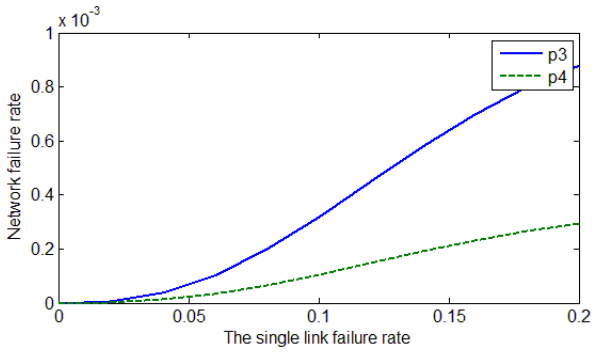


Fig 12 the comparison between P_3, P_4

In conclusion, the hot backup network through the network switch has the better reliability than other two operation modes.

VI COMPARING WITH OTHER SCHEMES

Besides the method introduced in the paper, there is another one for realizing the SpW hot backup redundancy [11]. That paper proposed that the main control node detects link fault in each sub network by sending RMAP package to read the link state of router, combined with the control of time-code reserved bit, achieving SpW hot backup redundant network. There are some deficiencies in that approach: (1) there is an additional work for the main control node, reducing the efficiency of data link and it is easier to cause link congestion. (2) Network monitoring and network switching function is not achieve by the router. It will make the network more complex, which is not good for the implementation of improving reliability of network.

The approach in this paper greatly reduce the workload of main control node since it does not require the main control node sending the RAMP packet to read the link state of router and determine the sending time-code. The removal of RMAP packets transmission also mitigates the burden of the main control node, improving the link utilization for network application.

VII CONCLUSION

The paper introduces a method to realize a hot back up redundant SpW network. It also improves the reliability of SpW network by combining the characteristics of SpW time code concept and SpW-D technology. Research and analysis show that the method is useful and effective and achieves the higher network usage efficiency in working state. When a fault occurs in the network, the system can respond to a variety of fault conditions through the corresponding measures, and recover the data transmission in the network, ensuring the accurate and reliable work of SpW network.

REFERENCES

[1] ECSS-E-ST-50-12C SpaceWire-Links, nodes, routers and networks[S]. Noordwijk, The Netherlands: European Cooperation for Space Standardization, 2008: 13- 14

[2] Steve Parkes Allison Bertrand Martin Suess Glenn Rakow SpaceWire-2011 Proceeding of the 4th International SpaceWire Conference[R], November 2011.

[3] Mao NingYuan. Research Of Key Techniques In SpaceWire Redundant Network [D]. Harbin, Harbin Institute of Technology ,2012

[4] Niu Yue Hua, Zhao Wen Yan. Design and analysis of a strong fault-tolerant on-board SpW bus network [J], Journal of computer applications 2014,34(9):2497-2500,2504

[5] Chen J, YangS, MeiH. Study and Implementation of SpaceWire Network Redundancy Technology Based on FPGA[C] Proceedings ofthe 6thInternational SpaceWire Conference. Dundee, UK: Space Technology Centre, University of Dundee, 2014:202-206.

[6] SpaceWire Time-Codes[S] ISWS, 2003.

[7] International SpaceWire Conference[R] Nara,November, 2008.

[8] SpaceWire-D Deterministic Control and Data Delivery Over SpaceWire Networks[R], April, 2010.

[9] Zhang Hao, Wu Jun, Zhang Chunxi. Design Of Backup Fault Tolerant Protocol For SpaceWire On-Board Network [J]. Computer Measurement & Control, 2015, 23(2):633-636.

[10] Chris McClements, Steve Parkes, Gerald Kempf, Pierre Fabry.SpW-10X SpaceWire Router User Manual[S],July,2008

[11] Zhuang Hong Yi, Implementation and analysis of backup router based on SpaceWire Time-Code[J], Computer Measurement & Control, 2016,1(39):510-5030

JUICE Time Distribution Protocol

SpaceWire Networks and Protocols, Short paper

Felice Torelli, Jørgen Ilstad, Giorgio Magistrati
European Space Agency - ESTEC
Noordwijk, The Netherlands
{felice.torelli, jorgen.ilstad} @esa.int

Abstract*— JUpiter ICy moons Explorer (JUICE) is the first ESA large-class mission aiming at the exploration of Jupiter and three of its largest moons, Ganymede, Callisto and Europa. The JUICE payload suite counts 10 instruments, 9 of which are interfaced with the On-Board Computer (OBC) and Solid State Mass Memory (SSMM) using a SpaceWire (SpW) network both for science data and command & control traffic. JUICE requirements favor the adoption of SpaceWire ECSS protocol standards in a layered architecture fashion. With regards to the time synchronization and distribution, the "High Accuracy Time Synchronization over SpaceWire Networks" draft protocol specification (developed within the SpaceWire Working Group) has been analyzed and tailored to meet JUICE needs. The paper presents the rationale behind the use of the SpW TDP, it's tailoring and concludes with lessons learned and recommendations towards an ECSS standard document.

***Index Terms*—SpaceWire, Networking, Time Distribution Protocol,**

I. Introduction

Jupiter ICy moons Explorer (JUICE) is the first L-class mission in ESA's Cosmic Vision Programme foreseen in 2022. The objective of the JUICE mission is the investigation of Jupiter and its icy moons, Callisto, Ganymede and Europa. It will address the question of whether possible habitats of life are provided underneath the surfaces of the icy satellites and also probes Jupiter's atmosphere and magnetosphere.

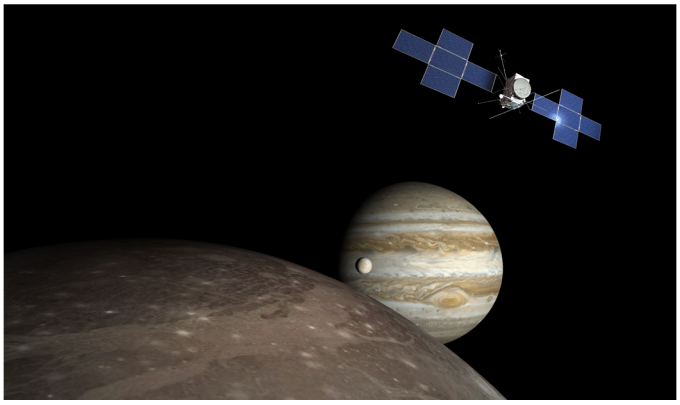


Figure 1: Artist impression of the JUICE spacecraft (courtesy of Airbus Defence and Space)

Ten instruments have been selected by ESA for the science JUICE mission. They can be gathered in three main categories: remote sensing instruments, geophysics instruments and in-situ instruments. Firstly the remote sensing instruments are focused on observation of Jupiter and its icy moons, their surfaces and the composition of their atmospheres. Then the geophysics instruments are dedicated to the restitution of the surface relief, the sub-surface composition and the gravity fields restitution. Finally in-situ instruments objective is to provide data on the Jovian environment mainly on the plasma and the fields surrounding the moons.

This paper focuses on the data handling subsystem between onboard computer (OBC), mass memory (MM) and payload instruments (P/L). Reliable and accurate time distribution and synchronization is important not only for spacecraft command and control but also to time tag samples of the payload instruments in a consistent and accurate manner. In this paper we discuss how the time synchronization is carried out between OBC and instruments over the SpaceWire network and how the SpaceWire Time Distribution Protocol (TDP), in line with ECSS SpaceWire protocol stack, has been applied as opposed to traditional approaches based on packet utilization standard (PUS) [05] private services.

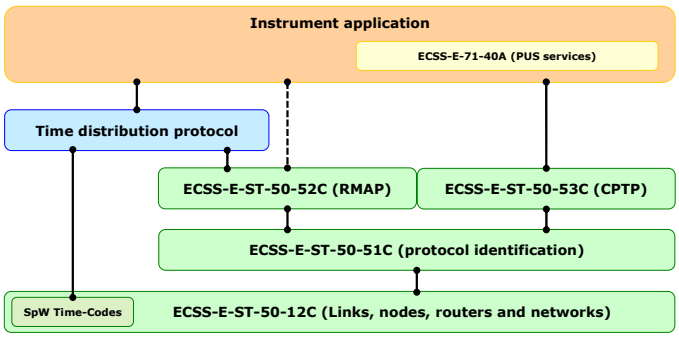


Figure 2 – SpaceWire protocol layers

The paper presents in section II how the time distribution and synchronization services are applied for the JUICE data handling system and in section III presents the detailed tailoring of the SpW TDP. In section IV a brief comparison is given between the traditional implementation based on the Packet Utilization Standard (PUS) custom made service 9 and the SpW TDP. Section V presents lessons learned and feedback from the

instrument developers on suggest improvements of the draft standard. Section VI presents the conclusions.

II. TIME DISTRIBUTION AND SYNCHRONIZATION

Time distribution (and synchronisation) is a function required by any intelligent instrument or unit and it is a task independent of the application-specific functions implemented by the instrument SW [08].

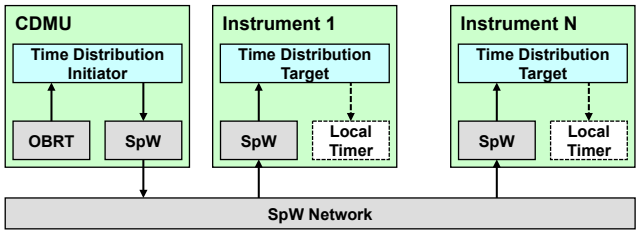


Figure 3: Time distribution context

The Command and Data Management Unit (CDMU) distributes periodically the On-Board Reference Time (OBRT) over the communication network, the instrument acquires the time and adjusts (i.e. resets) the local time to be in-synch with the on-board reference time of the CDMU. The on-board time distribution is managed by the CDMU, which is also the master of the OBRT and is in charge of :

1. distributing to the instruments (users) on the SpW network the value of the On-Board Reference Time (OBRT) being applied at the next synchronization pulse;

2. distributing to the instruments on the SpW network the synchronization pulse used to latch the OBRT, previously distributed, into the local timers of the instrument.

As a worst case, the period between two time synchronisation events is determined by the stability of the local timer part of the instrument. The accuracy of the distributed time is determined by the jitter and latency of the synchronisation pulse (e.g. SpW time-code).

In JUICE the draft standard "High Accuracy Time Synchronization over SpaceWire Networks" [10] has been adopted and tailored.

III. PROTOCOL TAILORING

The SpW TDP provides means to increase the accuracy of the system time by providing latency and time-stamp services that in principle achieves an accuracy of less than 100ns [14].

The accuracy for the telemetry timestamp of JUICE instruments, however, is not demanding. The requirements for JUICE in terms of jitter and latency is 0.5us and 5us respectively, henceforth there is no need to provide network latency compensation.

These values are by far higher than what the SpW network in JUICE can provide only in its basic form i.e. without adding compensation services provided by the protocol. Henceforth the protocol implementation can be "lighter" based only on this fact. It should nevertheless be stated that to utilise the high accuracy features of the SpW TDP protocol the SpW CoDec's themselves has to be able to accept and respond to distributed interrupts, a feature that will be introduced in the revision 1 of the SpW Standard.

One important consideration to take in to account as well, is that not all instruments support SpW RMAP hardware (HW) implementation. This requires the SpW TDP protocol services to be implemented in software. Thus, it puts limitations on what protocol features are possible to implement.

Many available LEON3 ASICs (GR712RC, EPICA-NEXT) include HW implementations of RMAP but don't support SpW TDP. HW support of RMAP is beneficial as it avoids complex protocol functions related to the RMAP to be implemented in SW.

Another consideration to take in to account when tailoring the protocol is that some platform units are FPGA-based i.e. not PUS terminals which allows the SpW TDP to be fully implemented in HW or SW.

The above considerations gave the incentive to tailor the SpW TDP as follows:

Mandatory services:

- Time Command Service (clause 5.2.4, [10])
- Initialization/Synchronization Service (clause 5.2.6, [10])

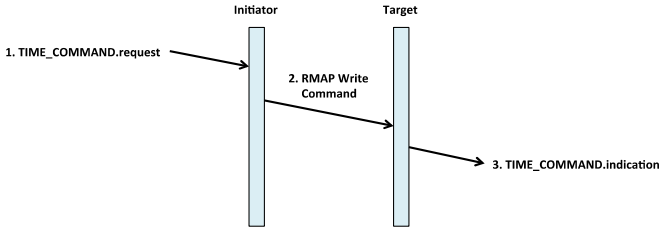


Figure 4: Time Command sequence

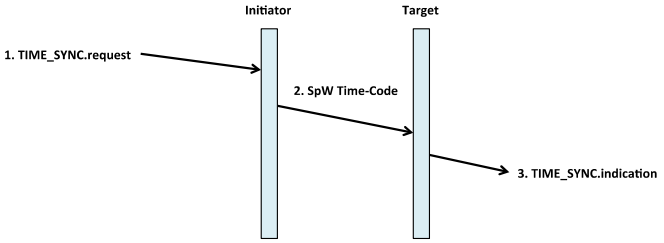


Figure 5: Time Synchronization sequence

Optional services:

- Status Service (clause 5.2.3, [10])
- Datation Service (clause 5.2.5 [10])

Hardcoded functions of the timing service:

- SpW time-codes are mapped on the LSB of *CCSDS Unsegmented Code - Coarse Time*
- SpW time-code distributed every second from OBC

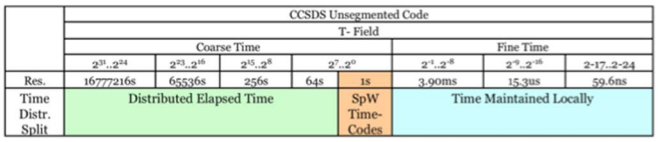


Figure 6: Format of the CCSDS CUC code.

In relation to the RMAP protocol, which the SpW TDP uses as foundation, the following minimal RMAP services are required:

- *Write non-acknowledged, non-verified to target* (clause 5.3, [03]).
- *Read command to target* (clause 5.4, [03]).

IV. SpW TDP and PUS 9

Prior to JUICE, some ESA missions customised ECSS-E-ST-70-41a [05] (PUS) service 9 for the implementation of a SpW time distribution protocol. This choice was also dictated by the lack of a dedicated sub-network protocol [08] that defined time distribution over the SpW network.

The current section provides a comparison between the custom PUS 9 "protocol" used in previous missions and the SpW TDP tailored for JUICE.

Flight heritage: At present only reference implementations of SpW TDP[13] exists, but no SpW TDP implementation has yet been applied in any space mission so far. Different flavors of PUS 9 have been used in previous science missions. Thus, from the perspective of flight heritage the adapted PUS service 9 could be considered to have more of an advantage.

Communication architecture: SpW TDP is in-line with recommended practices in terms of onboard data communication [11], [12], [13] and also provides adequate communication layering. While time distribution using PUS 9 mixes application layer (TC execution) and SpW protocol layer (time-code). From the perspective of communication architecture the SpW TDP is more advantageous.

Implementation complexity: SpW TDP requires one periodic RMAP write command (Time Command) and one periodic SpW time-code (initialization / synchronization). PUS 9 requires one periodic PUS TC (9, 132) and one periodic SpW time-code. From the perspective of implementation complexity they are considered equivalent.

Robustness: SpW TDP control field identifies which SpW time-code qualifies the distributed onboard reference time value (OBRT). With PUS 9, there are limited mechanisms to ensure that the OBRT received is not latched on the wrong time-code. This means the OBRT value in a particular time slot is latched anyway at the next received SpW time-code. From the perspective of robustness the SpW TDP is more advantageous.

Timing constrains: SpW TDP Time Command and SpW time-code can be processed immediately, typically right above the SpW SW driver. The PUS 9 TC has to be processed within the TC queue, which means its execution time depends on the instrument Application SW architecture. In relation to timing constraints the SpW TDP has a clear advantage.

HW implementation: SpW TDP is compatible with non-PUS terminals as it can both be fully implemented in HW and SW or a mix between the two, while PUS 9 can only be implemented in SW. From the perspective of HW implementation the SpW TDP is more advantageous.

Validation effort: SpW TDP requires "new" test procedures and validation tests to be developed, however there are less error conditions to verify as opposed using PUS 9. PUS 9 can rely on legacy test procedures and validation tests, but there are more error conditions to verify. From the perspective of validation the effort required is perceived as being equivalent.

The two protocols can be considered equivalent with regards to implementation complexity and validation effort. The custom PUS 9 has advantages in terms of heritage, however SpW TDP is more robust, it poses less time constrains for its execution, and it reflects an adequate layering of the instrument SW architecture. SpW TDP can also be adopted by both PUS and non-PUS unit on-board.

To summarise, the SpW TDP is advantageous if compared to the custom PUS 9 implementation and it should be the preferred choice in future missions where platform and payload units are connected by a SpW network.

V. Lessons Learned & Recommendations

In this section we present the lessons learned that have been collected so far as part of the JUICE development, aiming at providing suggestions for the improvement of the protocol document in anticipation of an ECSS standardisation.

1. The current draft protocol specification [10] lacks of a non-normative introduction that clarifies various use cases as well as provides architectural information about the protocol. Due to the missing non-normative part it is challenging to put the normative section in the right context
2. Only the essential bit fields are described in the draft protocol specification [10] and a number of TBD is still present. This fosters scepticism about the maturity of the protocol by the new adopters (e.g. instrument teams).
3. Mandatory and optional services are not identified in the draft protocol specification [10]. Because the full implementation of the protocol with all its features is unnecessary complex for missions where time requirements are not demanding, it is recommended to

better clarify which services and features that can be considered optional.

4. The protocol relies on SpW RMAP for the transport of commands and replies, without restrictions. Thus, although unlikely, the instrument memory integrity could potential be at risk in case an event causes an erroneous RMAP write command (used for Time Command) is sent and accepted by the instrument while in Science Mode.

Henceforth, the following recommendations should be considered included in the writing of the future ECSS standard document:

a. A non-normative section should present the context for the protocol to be used, use cases and a description of the services.
b. The bit fields and registers defining the protocol target memory should be fully specified.
c. A set of mandatory core services covering Time Command and Synchronization services should be clearly identified, leaving the latency compensation as optional.
d. While keeping the RMAP structure for the commands/replies, it is essential to limit the required commands to the bare minimum and to assign to the SpW time distribution protocol a dedicated Protocol Id, in order to confine the targets accessible memory/register area to the Time Distribution Protocol registers only.

VI. Conclusions

This paper has given an overview on how the SpW TDP has been adopted for the JUICE mission and also given the rationale behind the tailoring of the draft protocol description to fit the needs of the mission and the constraints the instrument developers are faced with. The simplification/tailoring allows a mixed HW/SW implementation.

The current draft protocol specification specifies no restrictions on the use of SpW RMAP for commands/replies and it has been strongly criticized by the users community (instrument teams). It is recommended that during the ECSS standardization process, this point is given particular attention, by e.g. constraining the use of RMAP for the SpW TDP protocol by assigning a dedicated PID.

VII. References

[01] ECSS-E-ST-50-12C, Space Engineering - SpaceWire – Links, nodes, routers and networks, July 2008

[02] ECSS-E-ST-50-51C, Space Engineering – SpaceWire protocol identification, February 2010

[03] ECSS-E-ST-50-52C, Space Engineering – SpaceWire - Remote memory access protocol, February 2010

[04] ECSS-E-ST-50-53C, Space Engineering – CCSDS packet transfer protocol, February 2010

[05] ECSS-E-70-41A, Space engineering, Ground systems and operations — Telemetry and telecommand packet utilization, January 2003

[06] CCSDS 851.0-M-1.0, Magenta Book, SOIS - Subnetwork Packet Service - Recommended Practice, December 2009

[07] CCSDS 852.0-M-1.0, Magenta Book, SOIS - Subnetwork Memory Access Service - Recommended Practice, December 2009

[08] CCSDS 853.0-M-1.0, Magenta Book, SOIS - Subnetwork Synchronisation Service - Recommended Practice, December 2009

[09] CCSDS 301.0-B-3, Blue Bool, Time Code Formats – Recommended Standard, January 2002

[10] SPWCUC-REP-0003, High Accuracy Time Synchronization over SpaceWire Networks, September 2012

[11] GR712RC User's Manual http://www.gaisler.com/doc/gr712rc-usermanual.pdf, April 2015

[12] SpW-18X SpaceWire Router GR718 Final Presentation 2014, June 2014

[13] SPWCUC-REP-0005 - SpaceWire Time Distribution Protocol - VHDL IP Core User's Manual, December 2013

[14] Spacewire time distribution protocol implementation and results, ISBN 978-0-9557196-5-3, ISPW 2014.

Distributed Storage System for Satellite Platform Based on SpaceWire Network

SpaceWire Missions and Applications, Short Paper

Niu Yuehua, Liu Weiwei, Wang Luyuan, Mu Qiang, Li Xin, Yu Junhui
Institute of Spacecraft System Engineering
China Academy of Space Technology (CAST)
Beijing, China
newjohn@126.com

***Abstract*—To satisfy more and more complicated requirements on information recording, processing, exchange etc. in newly developed intelligent satellites, a distributed storage system is designed based on SpaceWire network in satellite platform. In this distributed storage system, a SpaceWire router unit serving as the core device of star network connects several nodes, including onboard computers, namely, satellite management unit (SMU), data interface units (DIU) and other nodes like attitude and orbit sensors, space environment monitor instruments and payload controllers, etc. A certain number of standard flash memory modules are separately incorporated into SMU and DIUs to provide comprehensive data recording and exchange service for all nodes in the network. All memory modules are centrally controlled by SMU software to store authorized data in specific type files and retrieve data according to policies set by remote memory access protocol commands. Memory modules that can run in parallel in the distributed storage system produce very high data throughput, which boosts satellite platform capability to a great extent. Backup mechanism among memory modules is deployed and any module can be replaced by others when it meets failure, this makes the storage system robust and reliable. Testing and verification work performed on the distributed storage system shows that the system works quite well and also reveals its attractive effect on improving satellite platform efficiency.**

***Index Terms*—SpaceWire, network, distributed storage, satellite platform, memory module, throughput.**

I. Introduction

SpaceWire constitutes switched fabric with wormhole routing routers, and is widely used in spacecraft programs [1-2]. With flexible topology and simple interface, it offers network expansion and devices integration lots of conveniences. SpW data rate in range of 2Mbit/s-400Mbit/s as well as bidirectional full-duplex technology covers most transmission needs onboard except for some very high speed sensors. Thus SpaceWire is of notable advantage on satellite platform speedup and strengthening. As processing ability of onboard electronics grows quickly in the past few years, newly designed intelligent satellites deploy more complicated functions such as autonomous health management, mission planning and system reconfiguration [3-4], which demand satellite platform to support concurrent and deterministic data record and retrieval service for mass and miscellaneous onboard data. Conventional mass memory unit specializing mainly in high speed payload data recording and downlink data transmission with dedicated interface and protocol, can hardly adapt to the complex satellite platform applications. Nowadays modular avionics technology has been applied in spacecraft platform and flash memory modules are incorporated in onboard computers e.g. satellite management unit (SMU) and data interface units (DIU). Since the rate and capacity of single memory module is limited, a distributed storage system is designed based on SpaceWire network in this paper. In this newly-designed system, SMU, DIUs and other nodes such as attitude and orbit sensors, space environment monitor instruments and payload controllers, etc. are connected to the central router unit of star shape SpaceWire network, which provides real-time data transmission paths for all nodes in the network. A certain number of standard flash memory modules are separately incorporated in SMU and DIUs to provide comprehensive data record and exchange service for all nodes in the network. SMU software configures routing tables of SpaceWire network and controls work mode of all memory modules by remote memory access protocol RMAP commands [5], in this manner the network data flows are appropriately routed to and from memory modules based on balance and deterministic principle. Multiple memory modules in the storage system working together provide large capacity and high data throughput performance, and that is configurable in wide range by incorporating certain number of memory modules and setting flash chip number on single memory module. The backup and replacement strategy among memory modules make the distributed storage system robust and reliable, suitable for harsh space environment application.

II. Onboard Distributed Storage System

A. Drawbacks of Conventional Mass Memory

In satellites platform single solid state mass memory (SSMM) equipment is ubiquitously deployed. It connects to scientific payloads and onboard computer with dedicated input and output ports, through which high rate scientific data and low rate housekeeping data are collected. All recorded data is

only sent to formatter and encoder unit for downloading to ground station. The architecture and interfaces are designed to work in fixed mode without openness, which makes the system hardly accommodate different mission requirements with simple configuration, but often a large amount of redesign work. And because all recorded data can only be sent to ground for processing, onboard data indexing, retrieval and utilization cannot be supported, thus advanced onboard data processing and analysis tasks in new satellites are constrained.

As onboard processing ability has been enhanced in new satellites, the platform is required to provide more efficient and flexible data record services to support new tasks. In mass memory device aspect, standard interfaces and comprehensive functions are mainly concerned. And high speed bidirectional communication network that connects mass memory and user nodes together is expected at system level.

B. Distributed Storage System Architecture

SpaceWire has been widely adopted in satellite data management system to provide universal communication standard among onboard computer, solid state mass memory, payload instrument, etc. Many SpaceWire data system includes one SSMM equipment to fulfill all data record functions which indeed achieves high integration level, but on the other hand increases the equipment complexity. Multiple users' accessing to the common mass memory device also produces low efficiency problem. Furthermore, with one SSMM will take high single point failure risks such as power module breakdown, which may greatly affect the whole satellite mission.

The distributed data storage system is constructed based on SpaceWire network as shown in Fig. 1. Several onboard computers and other data processing nodes are all connected onto SpaceWire network and SpaceWire routers provide dynamic data transfer channels for all nodes. General mass memory modules are integrated in each onboard computer, and all memory modules on SpaceWire network form distributed storage architecture. This architecture achieves a number of advantages: 1) since onboard computer is designed in avionics modular manner, memory modules can be easily added into computer and communication between processor and Flash mass memory is simply accomplished on backplane bus. 2) Multiple memory modules in the system can work in parallel mode, this provide higher data throughput than single mass memory equipment. 3) Total memory capacity in the distributed system can be well customized by changing memory module number and memory capacity on each module without influence on other parts. 4) Single point failure is eliminated by distributing memory modules in different computers, making the system more robust.

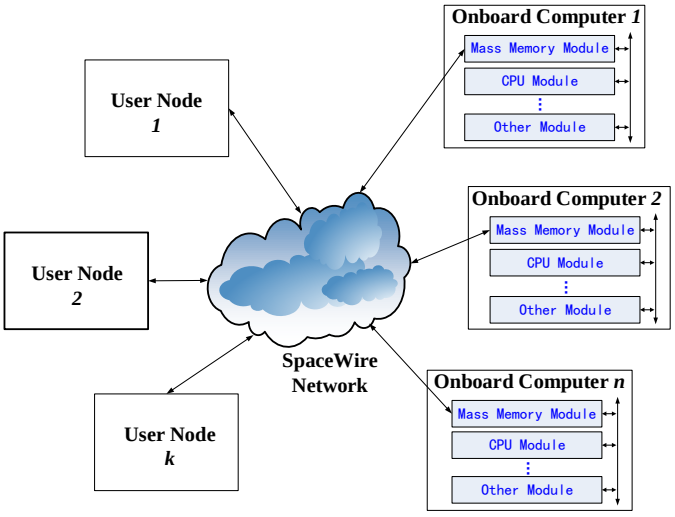


Fig. 1. Distributed storage system architecture

C. Integrated Data Services

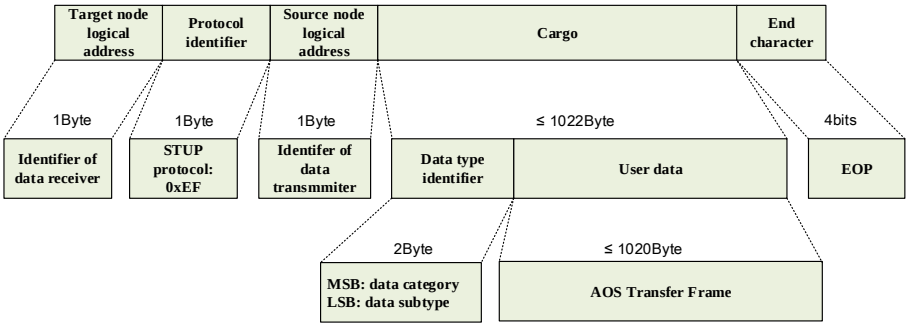
Storage system in satellite platform works as an infrastructure, providing interactive data service, downlink data service and uplink data service to satisfy diverse data recording and utilization needs onboard. Interactive data is recorded in mass memory modules after generation and transferred to onboard processing nodes on demand through SpaceWire network. Downlink data includes experiment data, engineering telemetry, event report, etc. These data are written into different mass memory modules with high resolution and downloaded to ground station for further analysis and health diagnosis. Uplink data including application images and patches, algorithm datasets, model library, etc., are sent to satellite from ground station through high speed upward channel. These data are firstly recorded in mass memory module installed in SMU and then forwarded to target user nodes in packet format through SpaceWire network. Due to the general interface of storage system, other data application mode can also be supported, e.g., downlink data being transmitted to onboard processor for preprocessing, uplink data being transferred on to 1553B bus for remoter terminal updating. When data is transmitted to low speed node, mass memory module in distributed system can tune packet interval to adapt, avoiding blockage in routers.

D. Data Transfer Protocol

Data transferred in the distributed storage system is categorized into two types. The main type is user application data that user nodes write into or read from mass memory modules. The other type is management data used for mass memory modules work mode control and state management, including commands of file operation like creation, deletion, close etc., or file write pointer, read pointer and other file information.

i) User Data Transfer Protocol

User data transfer protocol is designed by taking into account various factors: 1) Onboard data is primarily recorded, transferred and processed according to frame format defined by CCSDS standard. 2) Data write and read operation is based on flash page size in mass memory modules for high efficiency and low complexity. 3) Data flow in SpaceWire network is encapsulated into packets. Short packets reduce transfer efficiency, while very long packets increase network latency. Taking all these issues into account, the simple serial transfer universal protocol (STUP) is used to transfer user data in



Data Type	Category(MSB)	Subtype(LSB)
scientific data	0x00	0x00～0xFF User defined according to application
software image	0x11	
other data	TBD	
telemetry data	0xFF	

Fig. 2. User data transfer protocol packet format

distributed storage system [6]. Protocol packet format is shown in Fig. 2.

The three bytes length packet header consists of target node logical address, STUP protocol identifier and source node address, each occupies one byte. Packet cargo is composed of two bytes length data type identifier and a complete AOS transfer frame defined in CCSDS advanced orbiting system standard [7]. The fixed AOS transfer frame length is restricted below 1020 bytes and should be shorten when error correction encode is applied. Data type identifier is a tag for data cargo and is always recorded and transferred with AOS transfer frame. First byte of data type identifier is used to distinguish data according to main attribute, including scientific data, telemetry data, software image data, etc. Second byte is used as subtype identifier for each class of data. It is defined by data source node according to application needs. When a packet is received at a node, node application chooses data processing method according to data type identifier in the packet. In mass memory module data can be recorded in different files based on type identifier. At the end of protocol packet is an EOP character.

ii) Management Data Transfer Protocol

Management and configuration of mass memory modules and SpaceWire network is of significance in distributed storage system. Configuration and polling of SpaceWire router registers is implemented with general remote memory access protocol (RMAP), providing CRC verification and reply acknowledgement to assure reliable communication. Considering data reliability and commonality of application software design, the interface of mass memory module controller is similarly designed according to RMAP protocol. Data files management and access control are realized with RMAP command. File management commands include open, read, write, append, seek and close, etc. File access service support indexing data based on time or type, replaying data to SpaceWire node with specified logical address, inquiry of file information, etc. When mass memory module receives a RMAP command from SpaceWire network, command is verified and execution state is returned to source node, the result is also included in module telemetry.

III. Modular Onboard Computers

A. Hardware Architecture

Onboard computers are designed based on avionics modular architecture and a whole computer is assembled with standardized modules, such as processor module, mass memory module, telemetry module, pulse command module, etc. SMU and DIU are generated by integrating different group of modules according to functional demand. Modules in a computer communicate with each other through CPCI bus on backplane. Processor module acts as master node and is in charge of bus communication management, mode control and heath care of other modules. Processor module and mass memory module in a computer are regarded as independent smart node because both of them need to communicate with other SpaceWire nodes in the distributed system. Relying on the high speed backplane bus the two modules are designed to share one SpaceWire interface controller (AT7910), as shown in Fig. 3. SpaceWire controller is placed on mass memory module and directly connected to the FPGA with two extern

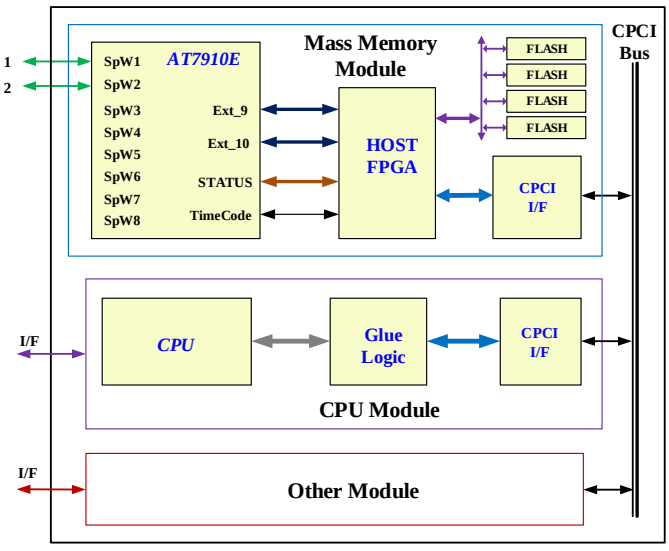


Fig. 3. Modular onboard computer hardware architecture

port EXT_9 and EXT_10. The FPGA interfaces with backplane CPCI bus and provide a bridge for processor application to access SpaceWire network. As CPCI runs at a rate of 32bit by 33MHz, it completely covers the demand from processor to SpaceWire network. This architecture enhances hardware efficiency and assures commonality of processor module, giving convenience to computer upgrade.

B. Communication Protocol

Processor module and mass memory module are independent nodes in SpaceWire network and each of them is assigned a unique logical address for packet addressing. Packets transferred between processor module and other nodes far in the network have to be forwarded by mass memory module FPGA controller. FPGA also deal with packets that processor applications write into or read out from FLASH memory on the module. To manage the four direction data flows, a simple small router is implemented in FPGA base on logical addressing principle, complying with SpaceWire standard. When FPGA receives packet from SpaceWire port it analyses the first byte logical address and direct packet to processor module through backplane bus interface or to internal control logic base on address matching result. If logical address is invalid, then the packet is discarded. The packets FPGA received from CPCI backplane bus are compatibly designed in SpaceWire packet format. When the first byte address points to mass memory module self, then packet is move to FPGA internal control logic or it is forwarded to SpaceWire port. With the uniform design of protocol format on backplane bus and distributed network, processor applications can easily access mass memory resources in or out the computer through one standard interface.

IV. Fault Tolerant Approach

Since storage system for satellite platform is required to be with high reliability, multilevel fault-tolerant approach is supported in this distributed storage system. By distributing several mass memory modules in different computers, namely, SMU and DIU, failure risks in the system are reduced. Each modular computer can accommodate more than two mass memory modules and provide redundancy and expansion ability, and multiple module parallel work mode produces better throughput and latency performance. Each mass memory module provides standard interface and is able to be substituted by other mass memory modules in case of failure or to replace any other faulty mass memory module. By dynamic configuration of SpaceWire network routing tables, data transfer channels to mass memory modules can be redirected, and specific RMAP command is design to update the logical address of mass memory module online, these approaches make the module replacement task easy to be accomplished.

SpaceWire network is the backbone of distributed storage system, all nodes, links and routers in the network are dual redundant. Network controller is implemented in SMU processor application to monitor network heath state and realize fault detection, isolation and recovery (FDIR). SpaceWire network routing tables are designed based on network physical topology and stored in application library. At startup, network controller initializes all routers in the network using data in the library. In operation period, it checks routing tables and other control registers in the network periodically. When a single event upset fault is detected, then the impaired register is scrubbed immediately. If a SpaceWire port is found abnormal it is firstly reset for recovery, if the abnormal state cannot be mended then the port will be disabled to avoid sending babbling data which may block the network. Watch dog timer is also enabled at each port on routers with a timeout setting of one micro second to prevent blockage caused by faulty nodes. The network controller is designed in cold spare mode and centralized control scheme based on network controller greatly strengthens robustness of the distributed storage system.

V. Test and Verification

A distributed storage system prototype is built based on star topology SpaceWire network in a remote sensing satellite platform upgrade program for test and verification, as shown in Fig. 4. The system includes three onboard computers, a SMU and two DIU, each computer is equipped with two mass memory modules, with 256Gbits capacity of each module and total capacity is 1.5Tbits. Other user nodes in the network include two payload simulator, a payload data processing unit (PDPU) and a data formatter and encoder unit (FEU). The whole network architecture is dual redundant and data rate is normalized to 100Mbps. SMU takes responsibility of network configuration and management. Engineering telemetry, housekeeping data and software images received from uplink channel are also recorded in SMU and SMU application commands these data to user nodes through SpaceWire

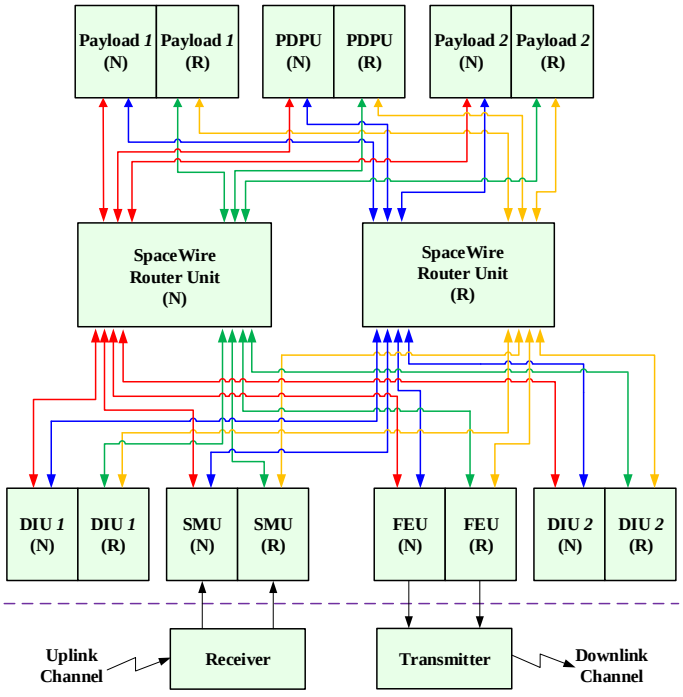


Fig. 4. Distributed storage system prototype for test

network. Two payload simulators work in parallel mode and generate experiment data at rate of 60Mbps and 40Mbps separately. The experiment data is sent to mass memory modules in the two DIUs separately. High speed payload data processing unit read experiment data from both DIU and request auxiliary data such as attitude parameters and orbit position, etc., from SMU to perform calculation and analysis. Then the result is written back into SMU mass memory module and downloaded to ground station with other recorded data in the storage system.

Data transfer protocol introduced in II (*D*) is implemented in the system, command and data are sent by applications simply based on target node logical address regardless of physical position of storage elements. The testing process and results have shown notable advance in system reconfiguration and expansibility. The distributed storage architecture well satisfies the needs of simultaneously processing of multiple high rate data flows, and the data transfer and processing ability of satellite platform is greatly boosted with SpaceWire network. The system is now planned to be applied in new remote sensing satellite missions.

VI. Conclusions

A distributed storage system is designed for intelligent satellite platform based on SpaceWire network, by incorporating flash memory modules in onboard computers. Central control scheme is used to provide concurrent, regular and balanced data record and retrieval service. Memory modules parallel operation and backup mechanism achieve high throughput, robustness and reliability. Verification result in a satellite program indicates the distributed storage system satisfies onboard data processing requirements quite well. Next step work will focus on more intelligent file transfer protocol in the system.

References

[1] European Space Agency, “Space engineering, SpaceWire - Links, nodes, routers and networks,” ECSS-E-ST-50-12C, Jul. 2008.

[2] S Parkes, P Armbruster. “SpaceWire: Spacecraft onboard data-handling network,” Acta Astronautica, 66, 88-95, 2010.

[3] Carlos G , Alan C, Gordon A. “Health management and automation for future space systems,” AIAA 2005-6803, Sept. 2005.

[4] Verfaillie G, Lemaitre M. “Tutorial on planning activities for earth watching and observation satellites and constellations: from off-line ground planning to on-ling on-board planning,” Proceedings of International Conference on Automated Planning and Scheduling, Cumbria, p29-39, 2006.

[5] European Space Agency, “Space engineering: SpaceWire - Remote memory access protocol,” ECSS-E-ST-50-52C, Feb. 2010.

[6] EADS Astrium GmbH, “STUP SpaceWire protocol specification,” SMCS-ASTD-PS-001, Jul. 2009.

[7] Consultative Committee for Space Data Systems, “AOS space data link protocol,” Blue Book CCSDS 732.0-B-3, Sept. 2015.

FDIR Method using an Embedded Timecode in Packets for SpaceWire-D

SpaceWire Networks and Protocols, Short Paper

Michiya Hayama
Information Technology R&D Center,
Mitsubishi Electric Corporation,
Ofuna 5-1-1, Kamakura, Kanagawa, 247-8501, Japan
Hayama.Michiya@db.MitsubishiElectric.co.jp

Hiroto Namikoshi and Isao Odagi
Kamakura Works,
Mitsubishi Electric Corporation,
Kamimachiya 325, Kamakura, Kanagawa, 247-8520, Japan
Odagi.Isao@cb.MitsubishiElectric.co.jp,
Namikoshi.Hiroto@bk.MitsubishiElectric.co.jp

***Abstract*— In satellite systems, triple modular redundancy (TMR) method with interconnected 3-CPUs is widely used to improve fault tolerance for the SEU/SET. Fault Detection, Isolation and Recovery (FDIR) functionality is also used to improve robustness of the system which isolates a faulty CPU and switches to a redundant CPU automatically. However, the FDIR does not work correctly in the following cases. First, SEU and SET may cause an unnecessary link occupation on the SpaceWire network. In this case, the voting mechanism and the fault detection mechanism work incorrectly due to the communication failure. Second, it is difficult to classify the cause of the fault combined with more than 1 failure mode by the master CPU. This paper proposes a novel FDIR method to overcome examples described above. The proposed method masks output signals of the SpaceWire interface with the error signal outputted from the voter. It enables the system to reset the link and notify the faults automatically. Furthermore, the CPUs notify each other the signal applying exclusive-OR (XOR) operation to the calculation results and a Timecode. This mechanism improves granularity of the fault classification. Finally, this paper clarifies the recovery time of the system in case of the double-fault including the link occupation by computer simulation. The simulation results show that the proposed method recovers the system with the same speed of the method which only uses a timeout mechanism.**

***Index Terms*—FDIR, SpaceWire, Timecode, Triple modular redundancy**

I. Introduction

The satellite payloads are required to ensure reliability and fault tolerance to the failure caused by SEU and SET. In general, TMR method is employed to them because it can reduce functional error with simple structure. For example, SpaceWire routers which uses majority voting with interconnected 3 CPUs through the network are proposed in [1],[2]. While, the multiplexing components through the network maintains the design flexibility, the network failure affects the FDIR functionalities. Because the notification of the faults and switching the redundancy depend on the network performance. In addition, it is difficult to classify the cause of faults only using the simple TMR mechanism. As a result, these systems cannot select appropriate recovery procedures based on the type of failure.

The SpaceWire is a standard considering an usage in the space environment and some works evaluated the effects of the SEU/SET in [3],[4]. These works targeted the faults of the single component. However, the report in [5] summarized 80% of the SEU in the satellite cause single-bit errors and the remained 20% cause multiple-bit errors to the SRAM in the space environment. This report suggests that the SEU causes failure in multiple components at the same time. In particular, if the components synchronize to the same Timecode (e.g. SpaceWire-D), whole system are affected by missing and delaying Timecode caused by the SEU/SET.

To deal with above mentions, this paper addresses high-reliability SpaceWire network by selecting the appropriate recovery method according to type of faults even if the multiple faults are occurred. This paper proposes the novel classification method which improves granularity of the fault classification for double-fault by embedding Timecode in the messages. Additionally, our proposal disconnects the SpaceWire links according to the classification result so as to enable each component to notify the faults to the other components. As a result, this mechanism can operate normally under the network failure. Finally, we clarify the classification granularity of proposed method by using failure mode and effect analysis (FMEA) considering the 15 types of the faults (e.g. occurring the SEU/SET, undeliverable Timecode). In addition, we evaluate the recovery time of the FDIR mechanism under the link occupation by using the computer simulation [6]. The evaluation result shows that the FDIR mechanism recovers the system with the same speed as an only using a timeout mechanism.

II. Embedded Timecode

This section describes how to embed the Timecode to exchanging messages. Figure 1 shows the block diagram of the proposed system. One of the 3 CPUs in the system operates as a master. The master collects the calculation results from the 2 other CPUs through the SpaceWire network then selects the

majority of the results. The voting mechanism is implemented in the Voter block. The Failure Mode and Effect Analysis (FMEA) block classifies the faults by analyzing the output signals from the Voter and the Update checker. The other 2 CPUs operating as a slave send the own calculation result with the CRC that is embedded a Timecode. Table 1 shows the encoding of the embedded Timecode. Each Timecode is encoded to the binary pattern either A or B. The CPUs select the binary patterns not to match previous result of the XOR operation. For example, the CPU selects the binary pattern B in case of the Timecode 4 because the previous XOR operation result is same as that of the pattern A. As a result, this mechanism detects an updating failure of the calculation results from the slaves by a simple comparison. Furthermore, the master can detect the difference of the Timecode between the other CPUs by checking the CRC which is recovered by applying XOR operation with the current Timecode.

Meanwhile, even though the master does not know Timecode pattern, not only Timecode patterns but also the CRCs decoded by each patterns have a relationship of bit inverting respectively. Therefore, the master can select the majority result using the circuit shown as Figure 2.

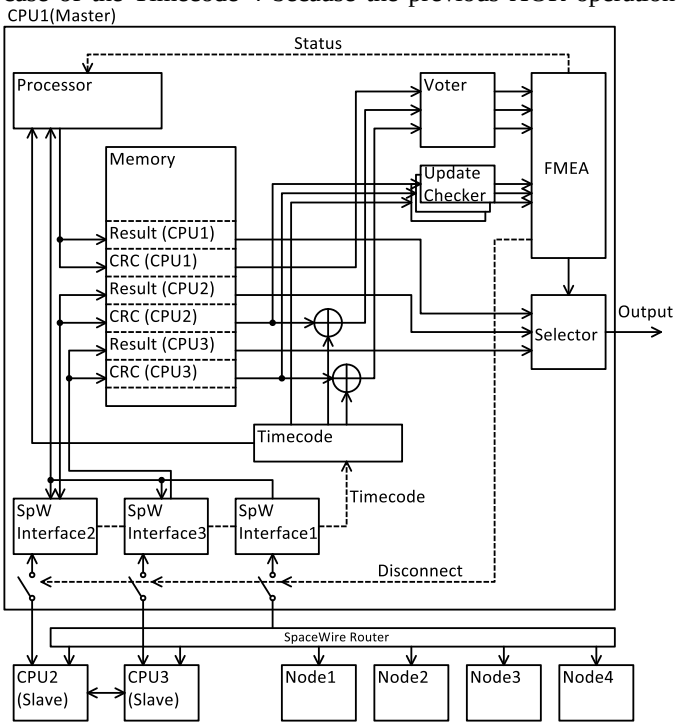


Fig. 1. The block diagram of the proposed system

TABLE I. THE BINARY PATTERNS OF EACH TIMECODE

Input	Timecode	Pattern A	Pattern B	Result
00000000	1	00000001	~~11111110~~	00000001
00001001	2	00000010	~~11111101~~	00001011
00001100	3	00000011	~~11111100~~	00001111
00001011	4	~~00000100~~	11111011	11110000
01000001	5	00000101	~~11111010~~	01000100
:	:	:	:	:

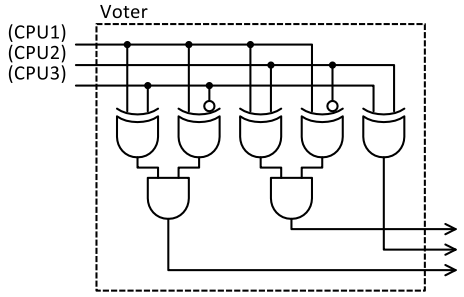


Fig. 2. The schematic of the voter

III. GRANULARITY OF THE FAULT CLASSIFICATION

This section shows granularity of the fault classification using the proposed method. The master CPU knows the information about the embedded Timecode and the comparison result of the corresponding calculation results from the other CPUs. These information improves the granularity compared to the simple TMR mechanism. Figure 4 shows that the number of failure modes combined with maximum 3 faults described in Table 2 are consisted in each input signal pattern of the FMEA block. The vertical axis indicates the input signal pattern whose meaning is shown in Figure 3. For example, it indicates that the result of the CPU1 and 2 is different with the result of the CPU2 and 3.

The result shows that the consisting failure modes in each pattern are increased with increasing of the combination of faults. If it considers the combination of 2 faults, the number of consisted failure modes is 5 at most except the one case (the input signal is "111000"). On the other hand, the number of consisted failure modes reaches nearly 30 if it considers the combination of 3 faults. These results shows that the proposed method is appropriate for the analysis considering the combination of 2 faults in maximum.

IV. THE FDIR METHOD

This section presents the FDIR method based on the analysis described in Section 3. We designed the FDIR method by defining the recovery operations using the FMEA tree considering the combination of maximum of 2 faults based on the analysis in Section 3. Figure 5 shows the FMEA tree.

According to the FMEA tree, the failure modes colored with gray (the input signal is "100000") mean the CPU1 or 2 might be failed. In this case, it is difficult to change the master to the CPU3 even though it is appreciate to recover the system when the CPU1 does not operate correctly. The FDIR method force disconnects the link between CPU1 and 3 in this case. This mechanism also releases the occupied SpaceWire link.

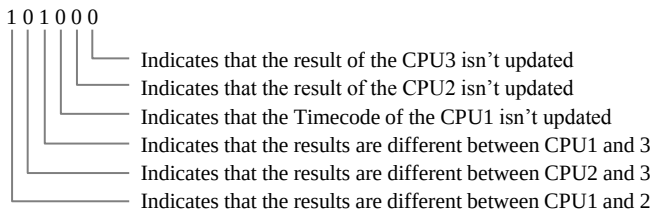


Fig. 3. The meaning of the FMEA input patterns

TABLE II. THE FAILURE MODES CONSIDERED IN THE ANALYSIS

Mode	Description
cpu1	CPU1 returned an invalid result.
cpu2	CPU2 returned an invalid result.
cpu3	CPU3 returned an invalid result.
time0	Timecode distribution failed (e.g. stopped the global Timecode distributor)
time1	CPU1 returned an invalid Timecode.
time2	CPU2 returned an invalid Timecode.
time3	CPU3 returned an invalid Timecode.
lost2	The result of the CPU2 was not delivered.
lost3	The result of the CPU3 was not delivered.
diff1	The updating checker for the Timecode in CPU1 returned a wrong signal.
diff2	The updating checker for the result of the CPU2 returned a wrong signal.
diff3	The updating checker for the result of the CPU2 returned a wrong signal.
voter1	The voter returned a wrong signal related to CPU1 and CPU2.
voter2	The voter returned a wrong signal related to CPU2 and CPU3.
voter3	The voter returned a wrong signal related to CPU1 and CPU3.

The failure modes indicated with bold box (input signal is "000010") means the failure of the detection circuit for updating results from the other CPUs and Timecode. In this case, CPU1 can notify the fault to the other CPUs using communication. Besides, the FDIR only stops the system ether when 2 or more CPUs failed or when the Timecode distribution failed in the most part of the system.

V. THE EVALUATIONS

This section shows the recovery time of the FDIR method described in Section 4 using the simulator based on NS-3. We evaluate the recovery time in case that both the link occupation described in [6] and the operation failure of the CPU are occurred simultaneously. The evaluation conditions are shown in Table 3. We compare two FDIR methods described in the following sections.

A. The Target Systems

1) The Conventional System

The system uses a simple TMR mechanism and the RMAP reply timeout for fault detection. The master CPU detects the faults including it self's one by the TMR mechanism. Then, it resets the SpaceWire link triggered by the RMAP replay timeout. The other CPUs take a master if it detects the disconnection of the link.

2) The proposed System

The system uses the FDIR method described in Section 4. The FDIR functionality in the master CPU recovers the system according to the FMEA tree shown in Figure 5 if it detects the failure. The other CPUs take a master if it detects the disconnection of the link same as the conventional system.

B. The Evaluation Result

The simulated packet trace of the proposed system is shown in Figure 6. The packet trace shows that the CPU2 operates as a master after the link disconnection which is generated by the FDIR functionality. The comparison of the recovery times is shown in Table 4. The results show that the recovery time of the proposed FDIR method is same as the conventional system.

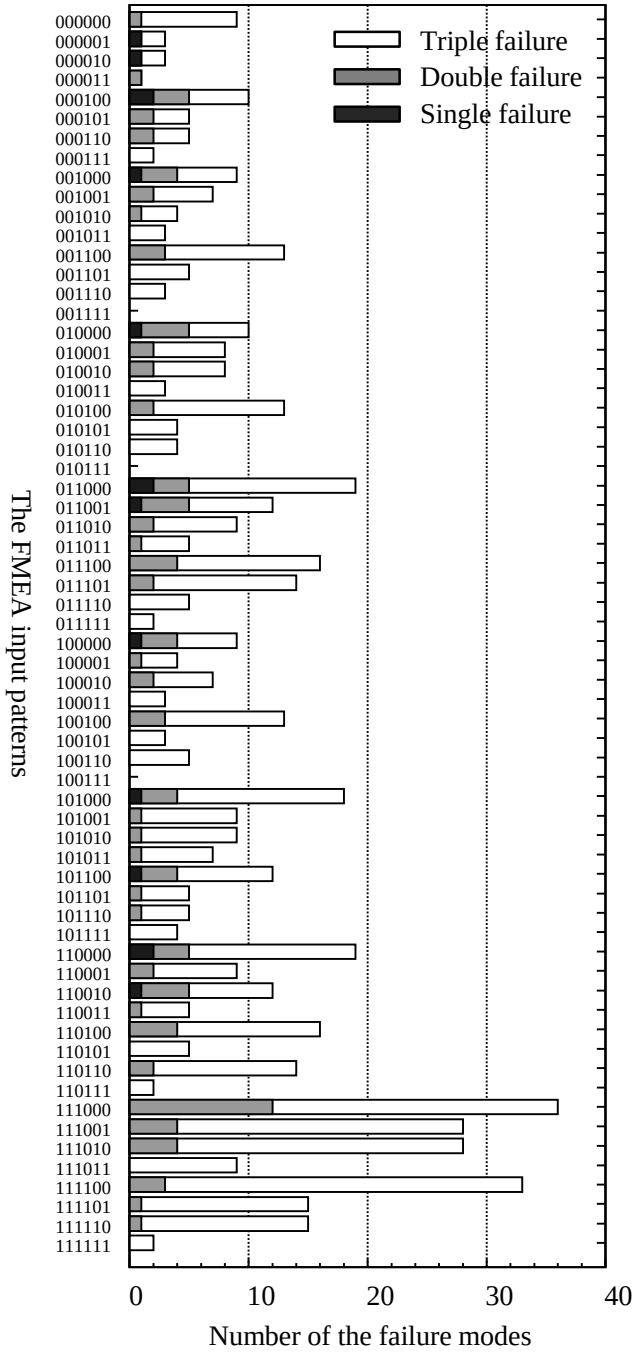


Fig. 4. The number of failure modes matched to each FMEA input pattern

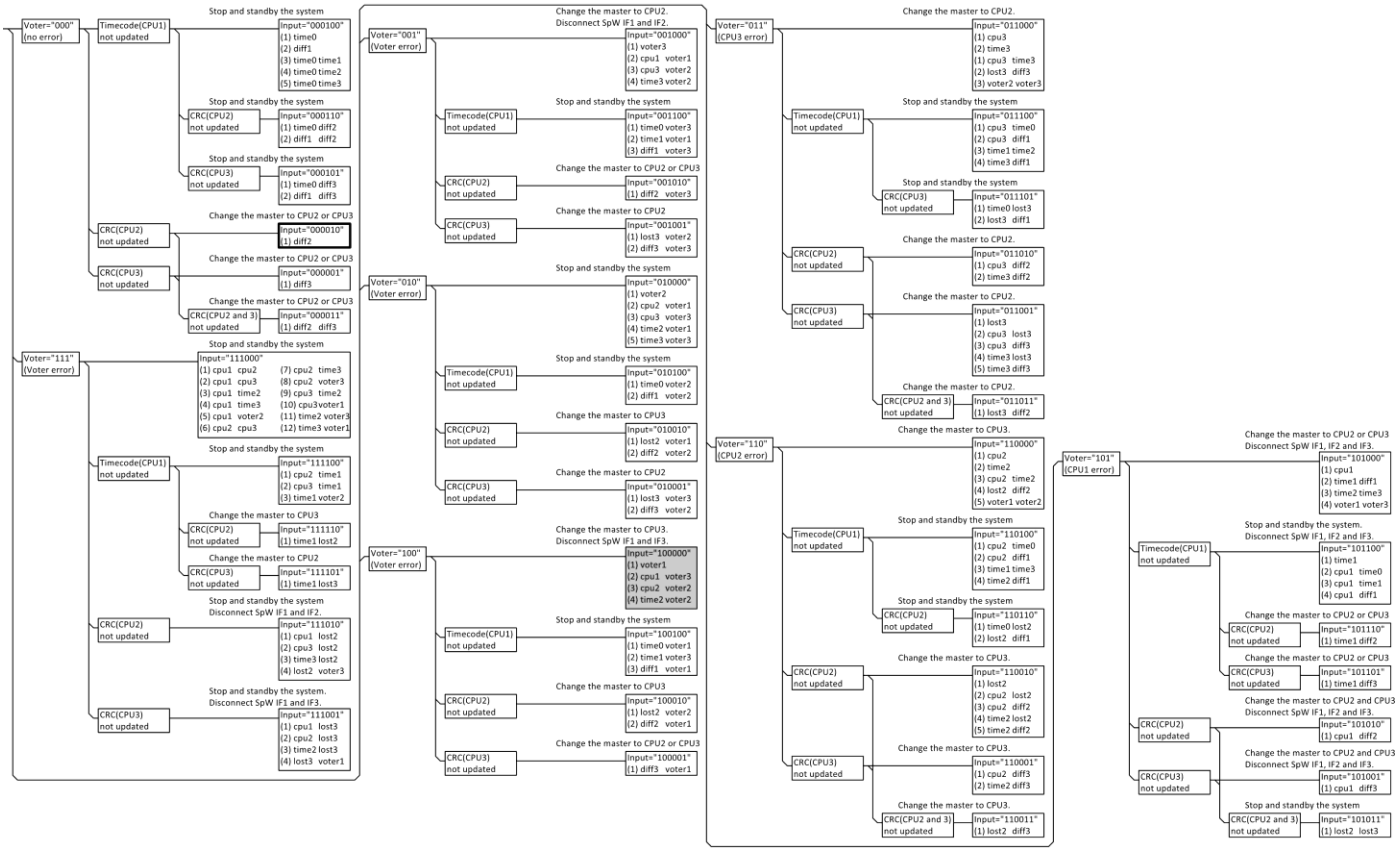


Fig. 5. The FMEA tree of the proposed system

TABLE III. THE EVALUATION CONDITIONS

Parameters	Conditions
Simulator	NS-3 with SpaceWire-D protocol[6]
Failure mode	Combination of following 2 faults. (1) A link occupation caused by SEU[6] (2) A calculation failure of CPU3
Slot	Interval time: 100 μs Number of slots: 8
Timing of counting votes	The next slot of collecting the calculation results of other CPUs
RMAP reply timeout	100 μs (equal to the interval time of a slot)
Data rate	10 Mbps
Network	3 CPUs, 1 Router and 4 sensor nodes

TABLE IV. THE SYSTEM RECOVERY TIME

Parameter	Conventional System	Proposed System
Recovery time	103μs	103μs

Because, the timing of counting votes in the proposed system is same as the end of the time for waiting the RMAP reply in the conventional system.

VI. CONCLUSION

This paper proposed the fault detection and classification method to improve granularity of the fault classification in the case of a double-fault by embedding the encoded Timecode to the exchanging messages between the 3 CPUs in the TMR system. We evaluated the classification granularity of the proposed method with considering the 15 types of faults. The evaluation result showed that the proposed method could classify the most part of the failure modes combined maximum of 2 faults into maximum of 5 failure modes. Furthermore, the proposed method can notify the faults to other components by disconnecting the SpaceWire links even if the link occupation is occurred.

In addition, we evaluated the recovery time of the FDIR method. The evaluation result showed the recovery time of the proposed method is same as the FDIR method using the simple TMR mechanism and the RMAP reply timeout.

REFERENCES

[1] A. Popovich, "Method Providing Fault Tolerance of Spacecraft Electronics by N-Modular Redundancy in Information Space of SpaceWire Network," Proc. of ISC 2010, pp.273-278, 22-24, Jun. 2010.

[2] F. Siegle, T. Vladimirova, J. Ilstad and O. Emam, "FDIR Techniques for Payload Streaming Applications using SpaceWire-based Networks," Proc. of ISC 2014, pp.11-18, 22-26, Sep. 2014.

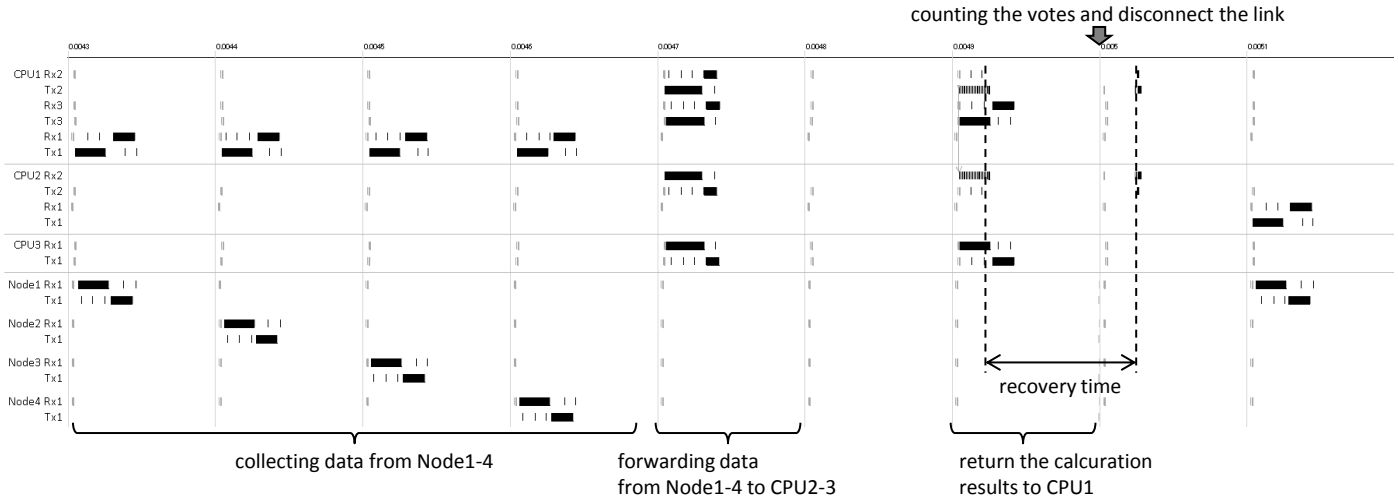


Fig. 6. The character trace of the proposed system

[3] Jimmy Tarrillo, R. Chipana, E. Chielle and F. L. Kastensmidt, "Designing and Analyzing a SpaceWire Router IP for Soft Errors Detection," Proc. of LATW 2011, 27-30 Mar. 2011.

[4] Jimmy Tarrillo, M. Altieri and F. L. Kastensmidt, "Improving Error Detection Capability of a SpaceWire Router IP," Proc. of RADECS 2011, pp.501-506, 19-23 Sep. 2011.

[5] Y. Bentoutou and M. Djaifri, "Observations of Single-Event Upsets and Multiple-Bit Upsets in Random Access Memories On-Board the Algerian Satellite," Proc. of NSS 2008, pp.2568-2570, 19-25, Oct. 2008.

[6] M. Hayama, Y. Yokoyama, R. Yagiu, I. Odagi and H. Namikoshi, "Impacts of Faults on a SpaceWire Network," Proc. of ISC 2014, pp.90-94, 22-26 Sep. 2014.

Generic ICU – A family of ICUs for MetOp-SG instruments

SpaceWire missions and applications, Short Paper

Alfonso Gonzalo Palomo
Space Equipment Engineering – Digital Engineering
Airbus Defence and Space - CRISA
Tres Cantos, Spain
alfonso.gonzalo@airbus.com

***Abstract*— Airbus Defence and Space Electronics have developed in last recent years a modular and scalable concept for Instrument Control Units (ICU) which has been used with high success in several missions as EUCLID and MetOp-SG. SpaceWire communication is a key element with this concept. This paper presents the Generic ICU architecture and its application in the MetOp-SG instruments.**

***Index Terms*— ICU, MetOp-SG, SpaceWire**

I. Introduction

The purpose of this paper is to present the Generic ICU concept developed by Airbus Defence and Space Electronics, the main constituents and the concept application to the different MetOp-SG instruments. SpaceWire is a key element in the architecture of the MetOp-SG satellite and it is widely used within the different instruments.

During the years 2013 and 2014 Airbus Defence and Space Electronics has conducted and internal R&D programme where the Generic ICU concept has been developed. The concept covers all the aspects related to a space electronics unit as modularity, scalability, mechanical and thermal design and electrical interfaces, both internal and external.

Variants of this concept have been applied in several units for EUCLID mission, as the Instrument Control Units for the NISP instrument or the Electronic Unit for the Fine Guidance Sensor Instrument. However, the full application of the Generic ICU concept has been performed in the MetOp-SG instrument with high success where 5 Instrument Control Units have been awarded to ADS Electronics during 2015.

II. GENERIC ICU CONCEPT

The electrical architecture of the Generic ICU is conceived as a cold-redundant one supplied by independent Main and Redundant Power Busses powering independent electrical chains, and managed by independent Main and Redundant TM/TC/Science SpW Interfaces with the platform.

The Generic ICU is composed by two sections with two main groups of functions/modules:

- A set of core modules which implements basic functions common to all ICUs, as processing functions, standard interfaces and power conditioning functions as Service DC/DC converter and Thermal control interfaces.
- A set of specific modules which implements very specific functions of the instruments as power distribution, especial interfaces and processing not covered by the core modules.

All these modules are connected together by means of a backplane and enclosed in a common mechanical housing.

The Figure 1 shows this concept:

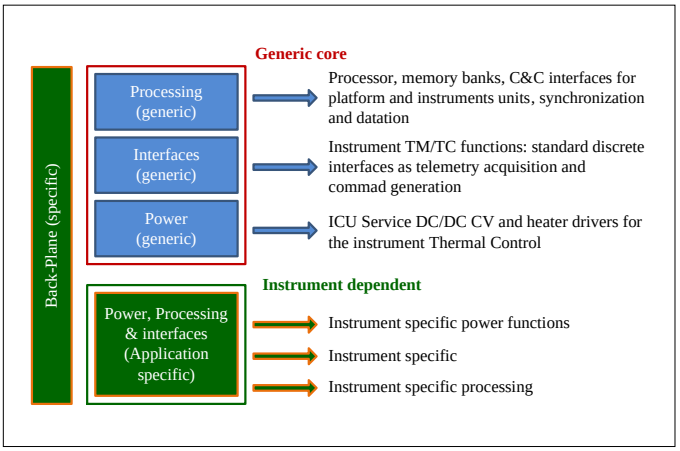


Fig. 1. Generic ICU architecture

A short description of each of the modules composing the general architecture is the following:

A. Generic Processing Module (GPM)

The GPM includes all the processing electronics (processor with its associated memory banks), the interfaces with the platform (**SpaceWire interfaces** with a router function, Equipment switch-off line –EQSOL- and synchronization interfaces), the interfaces with the downstream units (synchronization distribution, **SpaceWire** and UART links),

the interface to the internal backplane and the test interface. This module is based on a MDPA System-On-Chip ASIC (which includes a LEON-2FT microprocessor) and a RTAX2000 FPGA. The other ICU modules are controlled by the GPM by means of internal point to point SpW links, SPI links and discrete LVCMOS input/output lines.

B. Generic Interface Module (GIM)

It implements standard discrete I/O functions, such as generation of HLC discrete commands for power supply configuration of instrument downstream units as well as the acquisition chains of the thermal monitoring sensors (NTC & PTC types), voltages and bi-level status TMs provided by the units. This slave module is based on the Airbus DS's SECOIA ASIC (a general purpose board control ASIC) and is controlled by the GPM through an internal SPI link.

C. Generic Power Conditioning Module (GPCM)

It provides the power resources required by internal use of the unit (service DC/DC converter) and the thermal control support function, which in in charge of supply and control of the heater switches for the instrument operational thermal control function. Similarly to the GIM, this slave module is based on the same SECOIA ASIC and is controlled by the GPM through an internal SPI link.

In addition to these core modules, the ICU is completed by specific modules that implement all the functions and interfaces not covered by the core modules. The nature of this specific modules can diverse ranging from digital processing modules (for those instruments requiring data processing of digital video), or power conversion and distribution modules (for instrument requiring secondary power distribution to down-stream units), or a combination of functions of different nature in the same board.

All this modules are connected by means of a ***Backplane*** (***BP***), which provides basic power and interface services as follows:

- **Secondary Power lines:** the Service and Distribution DC/DC Converters in GPCM drive +28V/±15V/+8V/+5V secondary voltages towards the BP. The remaining modules use these secondary power rails for their internal needs and in case of additional voltages are required, they are generated by means of local regulation.

- **SPI busses**: the FPGA in the GPM manages two SPI busses through the MB, the SPI-1 for the TM/TC access to generic modules (GPCM and GIM), and the SPI-2 for the TM/TC access of specific modules, if needed. These buses are oriented to slave modules requiring low throughput (in the range of 1 to 10 Mbps)

- **SpW links:** the GPM provides two point-to-point SpW links through the BP for those modules requiring high data throughput (from 10 to 100Mbps). These SpW are connected to the SpW router in the GPM module which allows direct downloading of scientific data without software intervention.

- **Discrete lines:** they are managed by the FPGA in the GPM to transmit synchronization and command and control signals to the remaining slave modules.

The ICU is an intelligent unit with on-board software running on it. The GPM is in charge of hosting the Basic SW and the Application SW. The Basic SW has two main components:

- **Boot and Service Mode Software (BSSW)**: It provides basic services for GPM board monitoring and control (e.g. memory and registers load and dump…). It is also able to load and activate the instrument application software.
- **Execution Platform Software Package (EPSW)**: contains elements to build the instrument application software. This includes libraries to be linked to the application software (e.g. real-time kernel, standard PUS services, low level I/O drivers) and the associated tools (e.g. compiler, linker…).

The Generic core architecture is supported by two key ASICs developed by Airbus Defence and Space and available as recurrent products:

- **MDPA ASIC**. The MDPA ASIC is a System-On-Chip designed by Airbus Defence & Space GmbH and manufactured by ATMEL using the ATC18RHA technology. It embeds a LEON2FT microprocessor, which executes the processor's embedded SW, and interface control buses used on satellite – SpaceWire, 1553, UART – and with state of the art utilities for processor such as floating point unit, AHB bus, debug function, memory controller, etc. The MDPA ASIC is flight proven (TRL-9) on Alphasat Payload Controller since July 2013.

- **SECOIA ASIC**. The SECOIA ASIC (SErial COntrol Interface ASIC) is a general-purpose ASIC intended for communicating a slave electronic module with a master one in order to command and control their different internal functions (command generation, Bi-level / Relay status acquisition, Analogue Acquisition and serial interfaces). This ASIC has been developed by Airbus Defence and Space - CRISA and manufactured by Aeroflex using the ATC18RHA technology. This ASIC is widely used in many other Airbus DS products The device has been submitted to full qualification at component level and it has achieved TRL-8

In order to raise the TRL level of the Generic ICU concept, the most important modules of the ICU Core were developed through an internal R&D programme: The GPM (including HW design and Boot and Service Mode Software) and the GIM modules.

Next pictures show the EM models of the GPM and the GIM boards.

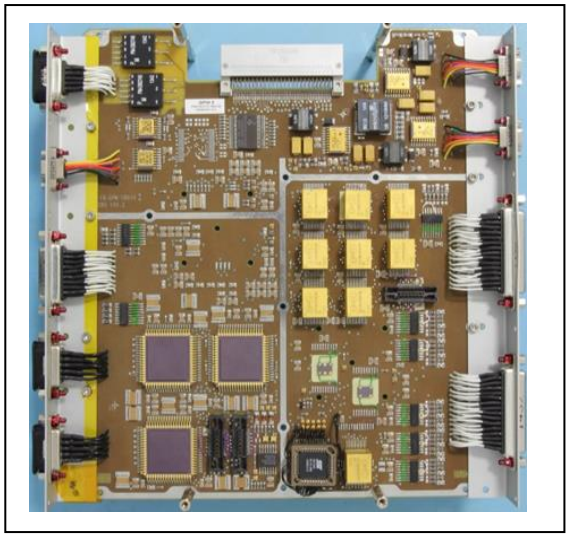


Fig. 2. GPM EM model

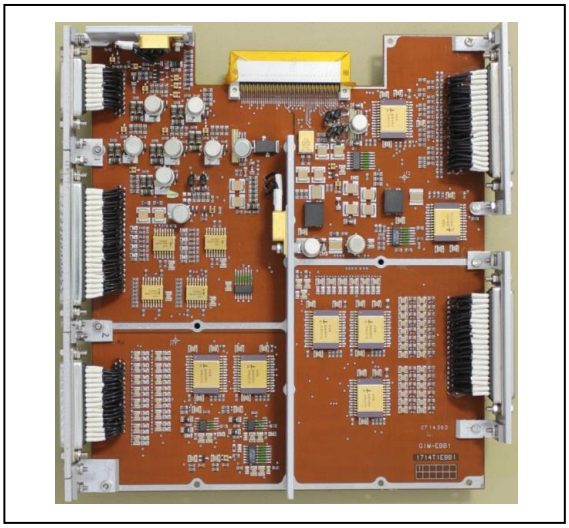


Fig. 3. GIM EM model

III. Generic ICU application in MetOp-SG instruments

The Generic ICU concept has been applied in the following MetOp-SG instrument control units.

- Sentinel-5 Instrument Control Subsystem (S5 – ICS).
- Microwave Sounder Instrument Control Unit (MWS – ICU).
- Multi-viewing, Multi-channel, Multi-polarization Imager Instrument Control Unit (3MI - ICU)
- MetImage Main Control Electronics (MetI – MCE)
- Scatterometer Digital Control Unit (SCA – DCU)

Main characteristics of the instrument and their ICUs configuration are provided following.

A. Sentinel-5 Instrument Control Subsystem (S5-ICS)

The Sentinel-5 instrument is a spectrometer covering the UV1, UV2/VIS, NIR, SWIR1 and SWIR2 ranges. The mission of the instrument is to detect the presence and local distribution of various gasses and aerosols in the earth atmosphere.

The ICU for Sentinel-5 is composed of the generic core (GPM + GIM + GPCM) plus 3 additional specific boards:

- Specific Interface Module (SIM). This module is in charge of providing interfaces with the instrument calibration subsystem.
- Specific Processing Module (SPM). This module is in charge of the acquisition and processing of the 5 spectrometer channels. Data generated by the SPM are routed to the P/F SpW link directly through the routers implemented in the GPM.
- Specific Thermal Control Module (STCM). It implements the thermal control function of the instrument for Survival and LEOP.

B. Microwave Sounder Instrument Control Unit (MWS – ICU).

The Microwave Sounder (MWS) is a 24 channel self-calibrating Microwave Radiometer. MWS provides the operational microwave humidity sounding capability for the METOP-SG meteorological satellites.

The ICU for MWS is composed of the generic core (GMP and GIM) arranged in a single board plus 3 specific boards:

- Signal Processing Electronics module (SPE). This module is in charge of providing high linearity video acquisition channels. Data are collected by the GPM, processed and formatted by the software for further transmission via the P/F SpW link.
- Rx power and Rx 54GHz secondary voltage switching Module (RxM). It provides a distribution DC/DC converter and switching matrix to the 54 GHz radiofrequency receivers
- Service Power and Rx non-54GHz switching module (PRx). It provides a second distribution DC/DC converter and switching matrix for the non-54 GHz radiofrequency receivers

C. Multi-viewing, Multi-channel, Multi-polarization Imager Instrument Control Unit (3MI - ICU)

The 3MI instrument is a wide-field of-view spectro-radiometer that is designed to acquire sequential images of the same ground target which are combined with multiple spectral views in both un-polarized and polarized channels in the VNIR and SWIR spectral ranges.

The ICU for 3MI is composed of the generic core (GPM + GIM) plus 1 additional specific board implementing the Service DC/DC CV and one additional distribution Converter for Front End power supply.

Front End data are acquired directly through the SpW links available in the GPM, processed and formatted by the software for further transmission via the P/F SpW link

D. MetImage Main Control Electronics (MetI – MCE)

MetImage is a passive imaging spectro-radiometer, capable of measuring thermal radiance emitted by the Earth and solar

backscattered radiation in 20 spectral bands from 443 to 13.345 nm (VNIR, SMWIR and VLWIR). The instrument achieves global coverage with 500 m square pixels.

MetImage Central Electronics (MCE) is composed of the generic core (GPM + GIM + GPCM) plus 2 additional specific boards:

- Rotation Control Module (RCM). In charge of controlling the instrument mechanisms, including position sensors. Encoder position telemetry is provided via an internal Spacewire link.
- Data Formatter Module (DFM). It is charge of generation of the rotation synchronisation of the instrument, acquisition of the video data from the Front End Electronics. Data are processed, packetized and transmitted to P/F SpW link directly through the routers implemented in the GPM.

E. Scatterometer Digital Control Unit (SCA – DCU)

The SCA is a real aperture C-band (5.355 GHz) radar with 6 slotted waveguide antennas, comprising 2 dual (H- and V-) polarised plus 4 single (V-) polarised antennas, which are accommodated on three roof-top-shaped antenna assemblies. It has to illuminate two at least 645 km wide measurement swaths on each side of the sub-satellite track.

The DCU for Scatterometer is composed of the generic core (GPM + GIM + GPCM) plus 1 additional specific board implementing the specific radar functions, including the radar timing signal generation, the chirp generation, mission data reception, A/D-conversion, processing, formatting and forwarding to the P/F SpW link directly through the routers implemented in the GPM.

The MetOp-SG ICU mechanical configuration ranges from small units composed by 4 boards assembled in a single housing with nominal and redundant sections in separated boxes (SCA-DCU), up to large units assembling 12 boards (N+R sections) in a single enclosure (Sentinel-5 ICS).

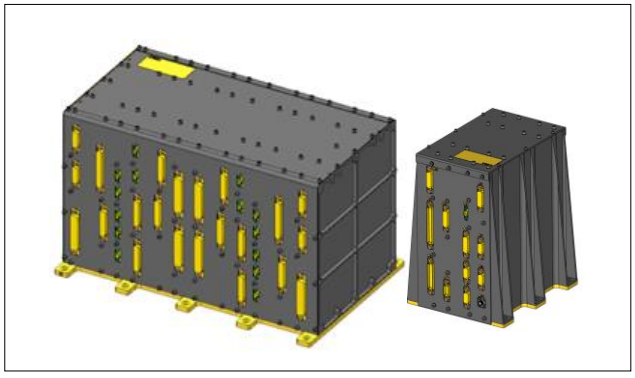


Fig. 4. MetOp-SG ICUs (S5-ICS –left- and SCA DCU –right-)

IV. USE OF SPACEWIRE IN METOP-SG SATELLITE

The MetOp SG Payload Data Handling System and Instrument communications are based on SpaceWire. MetOp-SG SpaceWire communications is based on the CCSDS Packet Transfer Protocol (CPTP) layered on the SpaceWire standard.

In addition, a Time Synchronization protocol based on SpW time-codes and CPTP packets is used to update the local time in each unit on the network.

Each OBC Processor Module is connected to the Mass Memory and Formatting Unit (MMFU) via two SpaceWire interfaces. All TC and TM data packet transfer to and from the MMFU is done via one of this SpaceWire interface.

All Instrument ICUs are connected by SpaceWire interfaces to the MMFU. Each Instrument has a dedicated (dual-redundant) SpaceWire interface for communicating mission data as well as monitoring & control data. The TM/TC packet routing and multiplexing is centralized within the MMFU.

The following picture shows the MetOp SG SpaceWire Network topology.

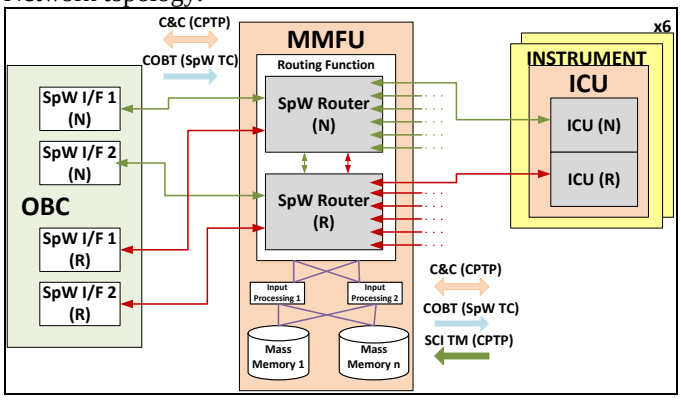


Fig. 5. SpaceWire Network topology in MetOp-SG

The SpW network is used for the following tasks:

- Command & Control (C&C) task from OBC to the MMFU routers and Payload Instrument based on CCSDS Packet Transfer Protocol.
- Transmission of the Central On-Board Time over the Spacewire network via a combination of SpaceWire time-codes and a PUS time update packet.
- Transmission of the scientific data from Instruments to MMFU for further downloading to ground.

V. USE OF SPACEWIRE IN THE GENERIC ICU

Management of the SpaceWire links in the GPM is performed by means of the MDPA ASIC. The MDPA processor incorporates two SpaceWire Modules with 4 ports each. Routing capabilities can be used within each module

The SpaceWire network at instrument level is depicted in Figure 6. Only one section of the ICU/instrument is shown.

Two network topologies can be implemented in the Generic ICU:

- Common network. Used for those applications where there the data processing and generation of the scientific data is

performed in real time with no intervention of the Software. Both SpaceWire Modules of the MDPA are connected together in such a way that a single network is implemented. Data volume generated by the instrument is very high and routing to the MMFU is performed by HW means. One of the links is used for Platform (P/F) connection; other link is connected to an FPGA. The other 4 links are available for external units and internal modules connection (slave modules within the ICU).

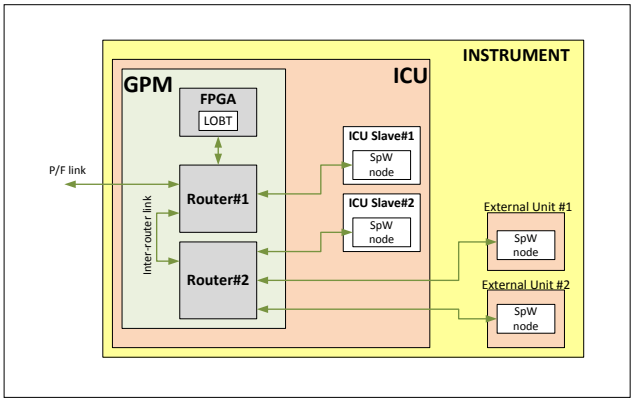


Fig. 6. SpaceWire Network topology at instrument level

- Two separated networks. Used for those applications where direct routing to the platform link is not required. Data processing and generation of the scientific data is performed by the instrument software. In some cases special management of the SpW link is needed, as generation of SpW time codes for synchronization purposes.

This topology is configured simply by not initializing the inter-routers links.

The ICU internal/external functions managed through the SpaceWire network are described following;

A. RMAP client in BSSW

The SpaceWire controllers in the MDPA don't support Remote Memory Access Protocol (RMAP) protocol by HW. A RMAP client has been implemented instead in the Boot Software which provides means for patch, dump and check the Boot EEPROM Memory through the P/F SpaceWire link without using the CCSDS Packet Transfer Protocol. Memory writing can be performed at ground level only.

B. Communication with the OBC (C&C, HK data)

The ICU is controlled and monitored via the P/F SpW link following CCSDS Packet Transfer Protocol layered on the SpaceWire standard.

Communication with platform is performed following an end-to-end data flow logical addressing scheme. The ICU is able to:

- Receive telecommand from the OBC
- Transmit housekeeping telemetry to the OBC
- Transmit ancillary data and housekeeping telemetry to the MMFU

All these transmissions are controlled at ICU level by the software embedded in the unit.

C. Instrument Local On-Board Time management

The ICU implements in the GPM FPGA a Local On-Board Time (LOBT) compliant with the CCSDS Unsegmented Code specification and is based on a default 32 bit LOBT Coarse Time field (indicating the number of seconds) and a 24 bit LOBT Fine Time field (indicating the sub-seconds).

The LOBT value is synchronized each a time tick message is received in the SpW port implemented in the FPGA. The datation function supports both LOBT direct and smooth synchronization methods.

Direct synchronization consists on LOBT setting upon reception of a SpW time tick with the COBT content which has been received via of a PUS time update packet.

Smooth synchronization is based in the same principle than Direct Synchronization but instead of a direct load of the LOBT, a computation of the LOBT drift with respect to the COBT is performed. This drift is used in the next 1 second counting cycle to compensate the excess or lack of count. In this mode, the Fine Time counter of the LOBT is never updated abruptly. This method allows also compensating the jitter and latency induced by the transmission of the SpW Time codes through the Spacewire network in the platform.

D. Scientific data transmission

Mission data are generated by the ICU either by specific HW processors implemented in the ICU or by Instrument's Application SW after acquisition from either external units or internal modules. Transmission of data to platform (either OBC or MMFU) is performed following a logical addressing scheme.

In case of HW generators internal to the ICU, the data are formatted as CCSDS TM packets, encapsulated into SpW packets and transmitted to the internal router where the node is connected to. The transmission to platform in this case is performed following a combination of path and logical addressing in order to route the information through the MDPA routers. First characters in the packet header are removed as long as the packet passes through one router to the other, in such a way that the first character at the P/F link is the logical address information.

In case of mission data generated by Application SW, the formatted packets are sent directly over the P/F link with the logical address at the packet header.

VI. CONCLUSIONS

The use of SpaceWire technology is a key element in the architecture of the Generic ICU. High speed data throughput as well as routing capability is an enabling factor for this high performant unit.

Simplicity of interfaces also allows offering a compact design from mechanical point of view with the consequent saving in mass and volume.

Deterministic Communication and Distributed Control of Avionics Based on SpaceWire-D

SpaceWire Missions and Applications, Short Paper

Liu Weiwei, Niu Yuehua, Cheng Bowen, Wang Luyuan
Institute of Spacecraft System Engineering
China Academy of Space Technology(CAST)
Beijing, China
akinglw@163.com

***Abstract*—As a switched network, SpaceWire can easily connect SpaceWire nodes together to realize parallel data communication, which can not only contain spacecraft interconnection between independent equipment, can also be used as "virtual backplane" to achieve mutual connection and communication between avionics internal hardware modules. However, SpaceWire nodes using asynchronous parallel operation mode are prone to causing network congestion, which is not conducive to the balance of network bandwidth. In this paper, a method of using the driver table between existing all-purpose interface controller of hardware module and SpaceWire node controller is implemented based on the SpaceWire-D protocol, to achieve deterministic SpaceWire network communication management. SpaceWire node send data to or receive data from the allowed nodes within the predefined time interval according to the information in the driver table, reaching the goal of deterministic communication and network flow optimization. In addition, through global communication time planning, when the processor module in certain avionics equipment fails while other hardware modules is still functioning, only the routing table in the SpaceWire router need to be reconfigured to complete the takeover of hardware module inside faulty equipment, without changing communication driver table information, thereby achieve the goal of distributed control.**

***Index Terms*—SpaceWire, Time-Code, driver table, virtual backplane, deterministic communication, distributed control.**

I. INTRODUCTION

With the continuous development of space technology, the electronic network interconnection technology from traditional low speed bus interconnections between subsystems gradually evolved in the direction of high-speed switched network [1]. SpaceWire provides extremely simple communication protocols, and its link rate can be tuned within the range of 2400Mbps, adapting to most of the instruments and equipment in spacecraft communications requirements. With the deepening of research and continuous improvement of the protocol, SpaceWire has been adopted by more and more spacecraft as the backbone of the data communication network, as the basis of implementing distributed avionics [2]. However, it is a pity that SpaceWire asynchronous parallel work between nodes and wormhole routing mechanism, makes the SpaceWire network's ability to adapt to the sudden data transmission is not strong, especially in the case of multiple link routing to the same destination, more prone to causing problem of network congestion and uncertain transmission delay [3]. Therefore, based on the SpaceWire-D, this paper implements a mechanism that all SpaceWire nodes are equipped with driver table for network communication management, and SpaceWire network is divided into multiple communication windows according to the information within driver table, which controls each node send to and receive from allowed nodes with specified length and within specified window. As a result, SpaceWire can mutually parallel communication without influence each other in one time window through global time planning. The communication driver table is executed and parsed by SpaceWire-D controller, which is increased table processing logic based on the original SpaceWire Codec core and RMAP core. According to the Time-Code values, SpaceWire-D controller can distinguish communication window and gets the corresponding control information in the driver table. If the communication window at this moment for receiving window and data updates, the all-purpose interface controller of hardware module will be notified for data reading and processing, if the communication window for sending window and data updates, data will be sent to the designated SpaceWire node. In addition, when avionics hardware module failure occurs in certain equipment, with the aid of the routing table of SpaceWire network reconfiguration and different time window planning for different hardware modules, the failed equipment can be taken over by other equipment, achieving smooth and seamless switching of data transmission and multitasking. This can also be adapted to the management and control of the equipment that without processor essentially.

II. DESIGN OF SPACEWIRE-D CONTROLLER

Figure 1 illustrates the structure of SpaceWire-D controller, which is composed by SpaceWire-D core and SpaceWire Codec core and RMAP core, and SpaceWire-D core is responsible for the management of communication window in SpaceWire network.

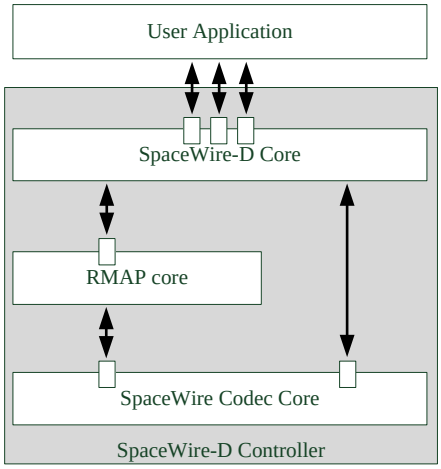


Fig. 1. Structure of SpaceWire-D controller

SpaceWire-D core not only able to communicate with the RMAP core, also can communicate with SpaceWire Codec core directly, to realize the RMAP packets and user-defined data packets transmission in SpaceWire network at the same time. By configuring different kinds of driven table in SpaceWire-D core can achieve communication window management, and can decide whether the SpaceWire network resources can be accessed and which packet type to be sent according to the information in table.

In order to guarantee the certainty of the parallel communication of SpaceWire network, all contents in the driver table are determined at the system design stage, cannot be changed during system operation, and cannot be accessed by user application. However, the whole SpaceWire system is allowed to carry different versions of driver table, used for system reconstruction and fault recovery.

There are two main driver tables in the SpaceWire-D core, which are ADDR_TABLE and SCHEDULE_TABLE as shown in Fig. 2.

The ADDR_TABLE is used to define the destination logic address of SpaceWire node that is allowed to access and the total length of sending and receiving data. The destination logic address and total data length can be obtained through simulation and calculation according to the time-slot and communication plan of whole SpaceWire network and the processing capacity of destination node. Besides, the table is divided into 64 segments that are corresponding to 64 time-slots respectively, and each segment contains LOGIC_ADDR field and TOTAL_DATA_LEN field.

- The LOGIC_ADDR field in each segment of the ADDR_TABLE is the destination logic address that is allowed to access. In each time-slot, the maximum destination logic address every source node allowed to access is up to six, and the six destination logic addresses within different SpaceWire nodes cannot be overlapped. If the actual number of destination logical address is less than six, the later corresponding position in the table is set to all zeros.
- The TOTAL_DATA_LEN field is the total length including sending and receiving data length, and is constrained by the node with the lowest processing performance, so in order to obtain higher efficiency of data communication, the destination node with different performance largely should be putted into different time-slot.

The SCHEDULE_TABLE is used to define the type of bus, multi-slot, the base address of the user memory space and the time-slot duration. This table is also divided into 64 segments corresponding to 64 time-slots in the same way, and each segment contains BUS_TYPE field, MULTI field, BASE_ADDR and SLOT_DURATION field.

- The BUS_TYPE field contains 3 bits. The value from 1 to 4 represents the static bus, dynamic bus, asynchronous bus, and packet bus respectively, while the value from 5 to 7 represents user-defined bus and protocol type, to send and receive data packets through the SpaceWire-D core communicate directly with the SpaceWire Codec core.
- The MULTI field is used to indicate whether using multi-slot. It contains 2 bits and the value represents the number of included time-slot of multi-slot.
- The SLOT_DURATION field contains 16 bits, which is used to define the duration of the time-slot. Each SpaceWire node use local time counter to divide time-slot and use the Time-Code to synchronize with global time of SpaceWire network.
- The BASE_ADDR field contains 11 bits, and is responsible for addressing the stating address of user control memory. Once obtaining the base address, the data in user data memory can be sent in the time-slot according to the information in user control memory just as whether or not the data is updated. The data format of user control memory as shown in Fig. 3.

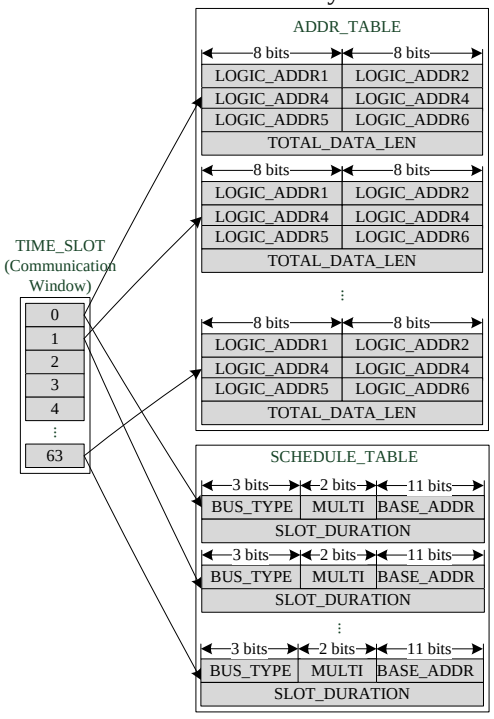


Fig. 2. Driver table in SpaceWire-D controller

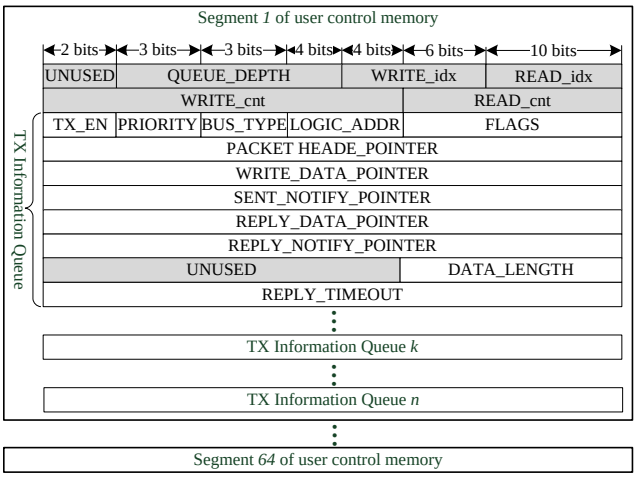


Fig. 3. Structure of user control memory

The user control memory contains 64 segments, each segment is actually an information queue to support transaction group, and the head of each segment provides the reading and writing information of TX information queue. In addition, each segment is divided into accessible spaces and inaccessible spaces; the accessible space in the Fig. 3 is represented by white areas, while the inaccessible space is represented by the shaded area. The accessible space mainly provides the information of data update flag (TX_EN), priority, bus type, destination node logical addresses, and the memory address pointer and data length and other information that RMAP core or user-defined protocol required.

When a certain time-slot comes, SpaceWire-D controller get the information in user control memory, and judge whether the data is updated, the bus type is same as indicated in associated index of SCHEDULE_TABLE, the logic address exist in ADDR_TBLE, and finally, whether the total data length is less than the specified maximum length in the ADDR_TBLE. When meet the above conditions, the RMAP core or SpaceWire-D core itself start data transmission depending on the bus type and address pointer. If there are multiple data update region in TX information queue, the static bus and dynamic bus read user control memory sequentially to transfer data, while the asynchronous bus read information in accordance with the highest priority.

It is important to note, SpaceWire-D controller does not distinguish between current transaction group and next transaction group for the reason that each time-slot corresponds to a separate base address. The current transaction group and the next transaction group can be associated with different base address, and the user application can choose to visit which base address of user control memory according to the current transaction group and next transaction group. For example, the time-slot 0 and time-slot 10 are all assigned to dynamic bus 1, the base address of time-slot 0 and time-slot 10 are set differently, and the current transaction group and the next transaction group can be realized by access two base addresses by turns. The base address associated to time-slot 0 and time-slot 10 can also be set to the same, and only the current transaction group is realized.

III. Construction of Spacecraft Avionics Communication Network Based on SpaceWire-D

The SpaceWire-D controller is used as the bus interface unit of the hardware modules to connect with the SpaceWire, and realize the data communication between the host controller (all-purpose interface control or CPU) of hardware and SpaceWire network.

A. The Composition of Spacecraft Avionics

The design of spacecraft avionics follows module and open structure [4]. Based on the analysis and abstract of system function, the nine standard and generic hardware modules with independent design and test capacity and a series of software component have been designed, as shown in Fig. 4. These hardware module and software components as the underlying fundamental support and under the framework of avionics standard bus architecture, the avionics equipment (management unit) with specific features can be designed by assembling different hardware module through internal communication bus, and to establish a complete avionics by aid of connecting every management unit through external bus [5].

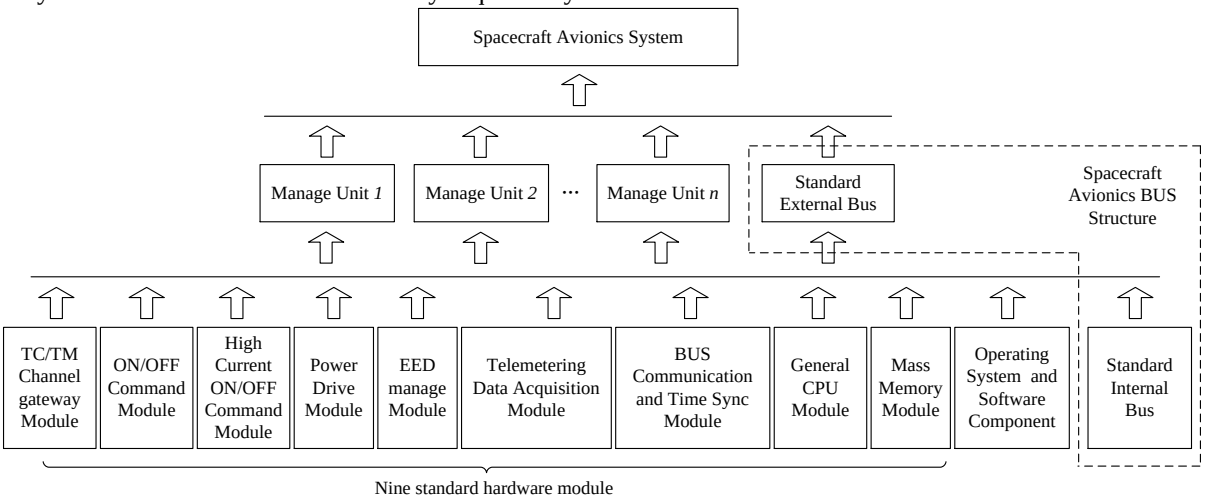


Fig. 4. Hierarchy of spacecraft avionics system

B. Application of SpaceWire-D Controller in Avionics

SpaceWire has been adopted as the internal communication bus between hardware modules within avionics management unit, and the SpaceWire network as the "virtual backplane" of equipment, to realize the interconnection between each hardware module through SpaceWire-D controller. In addition, besides the general CPU module, other hardware module configures all-purpose interface controller to control SpaceWire-D controller and as the communication bridge between SpaceWire-D controller and functional circuit. The all-purpose interface controller and SpaceWire-D controller are the uniform configurations of hardware module, while functional circuit is varied depend on the function of hardware module.

The external communication bus between management units also chooses SpaceWire. This makes the avionics with the same structure and interface, internal and external bus with the same communication protocols and access mechanisms. As a result, there is no level and grade difference and can achieve distributed and parallel operation between avionics equipment, and more important, this makes the avionics has the capacity of task migration, system reconstruction, coordination and cooperation among different equipment. The schematic diagram of the avionics communication network based on SpaceWire-D is shown in Fig. 5.

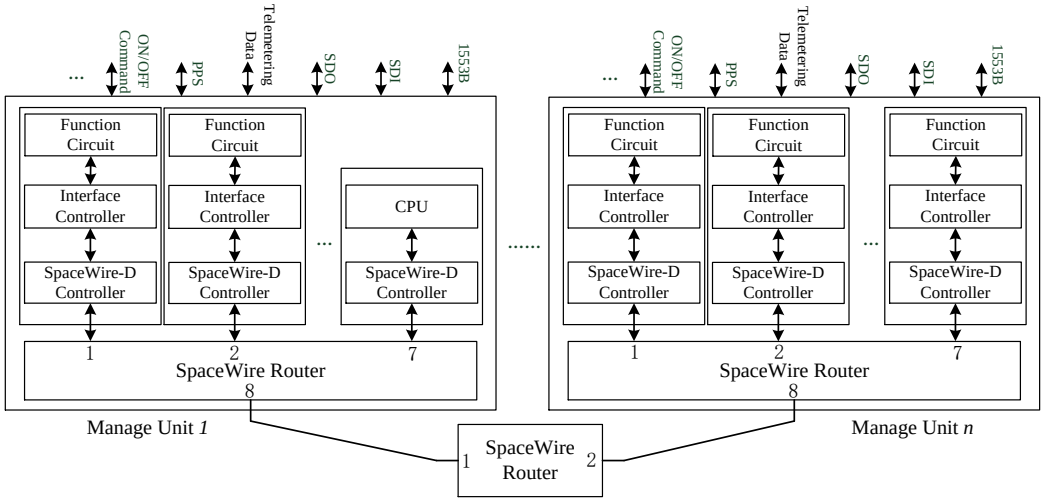


Fig. 5. Example of avionics communication network based SpaceWire-D controller

IV. Deterministic Communication and Distributed Control of Avionics based on SpaceWire-D

Relying on the time planning of SpaceWire-D and unify interface of the hardware module, makes it easy to realize deterministic communication and distributed control of avionics.

A. Realization of Deterministic Communication in SpaceWire Network

The communication network based on SpaceWire-D with global time planning characteristics, SpaceWire node and packets according to the arranged time-slot for transmission, there is no link resource competition and conflicts to ensure packet transmission latency and deterministic communication.

In order to ensure no link resource competition and collision in any time-slot, any target node only allows one initial node to access. Take the avionics communication network in Fig. 5 for example; each management unit contains seven hardware module, and two management units through port 1 and port 2 of SpaceWire router for connection. Each hardware module is numbered by the connection port of the router, and the number of target node that initial node allowed to communicate is six in one time-slot. The time-slot planning schematic is shown in Table 1.

TABLE I. EXAMPLE OF TIME-SLOT LAYOUT

Time-Slot	Initial Node	Target Node
0	1-1	1-2,1-3,1-4,1-5,1-6,1-7
	2-1	2-2,2-3,2-4,2-5,2-6,2-7
1	1-2	1-1,1-3,1-4,1-5,1-6,1-7
	2-2	2-1,2-3,2-4,2-5,2-6,2-7
2	1-3	1-1,1-2,1-4,1-5,1-6,1-7
	2-3	2-1,2-2,2-4,2-5,2-6,2-7
……	……	……
7	1-1	2-2,2-3,2-4,2-5,2-6,2-7
8	2-1	1-2,1-3,1-4,1-5,1-6,1-7
9	1-2	2-1,2-3,2-4,2-5,2-6,2-7
10	2-2	1-1,1-3,1-4,1-5,1-6,1-7
……	……	……

Because of the demand to avoid competition of link resources, the hardware module that directly connect a router can communicate simultaneously (such as time-slot 0, 1, 2), but when the hardware module needs to communicate cross the router, the communication have to be done in different time-slot (such as time-slot 7, 8, 9, 10).

Compared to traditional SpaceWire network, SpaceWire-D not only can improve the deterministic communication, and data package is no collision and competition between each other, also can according to the time-slot where packet error, to realize SpaceWire network fault location, isolation and restoration expediently.

B. Distributed Control in SpaceWire Network

Thanks to the internal bus and external bus of avionics equipment also use SpaceWire, breaks the boundaries between equipment on the logic, SpaceWire network as "virtual backplane", each hardware module can be seen as independent SpaceWire node, this makes the communication within and between management unit can be unified to design.

During the normal operation of the avionics system, different management units can operation independently and concurrently, and the data interaction between different management units is the processed data rather than the original data and only through general CPU module, reducing data traffic and improving the efficiency of communication.

When a failure occurs in general CPU module within certain management unit, the general CPU module within other management unit can directly take over the task and function of faulted general CPU module, to control and communicate with the hardware module within faulted equipment for task migration and system reconfiguration.

During the system reconfiguration and task migration, there is no need to change the configuration of driver table, but just reconfigure router table to make the logic address of fault general CPU module can be routed to the migrated general CPU module. However, there may appear a plurality of hardware modules to transmit data to a general CPU module in the same time-slot and need to reserve bandwidth when design SpaceWire network and SpaceWire-D driver table. There is another way to replace the destination logic address of the RMAP packet from the fault general CPU module to the migrated general CPU module directly, this need to provide corresponding mechanism to control all-purpose interface controller using new logic address when generating RMAP packet. This mechanism maybe implemented in sequent design.

V. FOLLOW-UP WORK

Although the SpaceWire-D can bring more benefits for the network communication, it still faces some difficulties in the process of network design, which will be optimized and improved in the following work.

- The time-slot planning is mainly dependent on manual work at present, which is very difficult and even cannot be completed when network contains more SpaceWire node, so it must to design simulation or calculation software to achieve the optimal timing planning.
- In the current design, the Time-Code have to be broadcasted frequently when time-slot duration is small, and cause the waste of network resources. Therefore, the future design will use local clock to generate multiple time-slot between two Time-Code, and Time-Code only to synchronize local time with global time.

VI. CONCLUSIONS

Through the SpaceWire-D realized the deterministic communication and distributed control, not only reduces the difficulty of avionics controller real-time multitasking, and enhances the system's ability to tolerate failure with the help of distributed control and faulty recovery. This advances the utilization of fault equipment hardware modules and the realizability of task migration, the robustness of switched network data communication and processing also be improved significantly.

REFERENCES

[1] DI Suran, ZHANG Weigong, CHEN Chuan, BA Feng, ZHOU Huazhang, WANG Jiajia. "The Development of SpaceW ire Simulation System Oriented Onboard Data M anagement System," MICROELECTRONICS& COMPUTER, vol. 29, pp. 164-167, January 2012.

[2] XU Shuqing， WANG Zhen， DONG Yaohai， LI Qing. "Research and Development of SpaceWire and SpaceFibre High Speed Bus," AEROSPACE SHANGHAI, vol. 31, pp. 29–36, 2014.

[3] YANG Zhi, LI Guojun, YANG Fang, LIU Shengli. "Design of Communication Protocol for SpaceWire On-Board Networks," Journal of Astronautics, vol. 33, pp. 200–209, February 2012.

[4] ZHAO Heping. "To build a highway to spacecraft intelligentization with avionics technology," Spacecraft Engineering, vol. 24, pp. 1–6, December 2015.

[5] Yang Tao, Li Chengwen, Yang J unxiang, Gao Yang, Wang Chunwei, Liu Yu. "Design and Application of the Mass Memory Module for Distributed Computer System," Computer Measurement&Control, vol. 22, pp. 909–911, 2012.

Network Latency Analysis of a SpaceWire-based Control System for Space Robotic Arm

SpaceWire missions and applications, Short Paper

Giuseppe Montano, Marek Rucinski *(Authors)*
Data Processing Advanced Studies Group
Airbus Defence and Space Limited
Stevenage, United Kingdom
giuseppe.montano@airbus.com
marek.rucinski@airbus.com

Elie Allouis[1], Olivier Notebaert[2], David Jameux[3]
(Co-Authors)
[1/]Airbus Defence and Space Limited, UK
[2/]Airbus Defence and Space SAS, France
[3/]European Space Agency, The Netherlands

***Abstract*— The Lightweight Advanced Robotic Arm Demonstrator (LARAD) is a state-of-the-art, two-meter long robotic arm for planetary surface exploration currently being developed by a UK consortium led by Airbus Defence and Space Limited under contract to the UK Space Agency (CREST-2 programme). LARAD has a modular design, which allows for experimentation with different electronics and control software.**

As ESA is drafting plans to address the design of sample return missions from key locations such as a Mars or Phobos, a number of technology developments are being undertaken across Europe to raise the maturity of key enabling systems, such as sample handling and robotic manipulators. One of these technologies is a high-TRL, SpaceWire-based control system for robotic arms currently being investigated by Airbus Defence and Space. This paper presents the results of a worst-case latency analysis of the fully SpaceWire-based control system currently being developed for LARAD. Some of the results are general enough to be extended to other robotics applications.

***Index Terms*—SpaceWire, Latency, Robotics, Robotic Arm, Planetary Exploration.**

I. Introduction

The control system of the Lightweight Advanced Robotic Arm Demonstrator (LARAD) is currently based on a combination of two communication protocols, Ethernet and CAN. The bandwidth limitations of the CAN protocol (maximum 1 Mbps, half duplex) have led to the need for the development of a new control system architecture for LARAD fully based on the SpaceWire protocol. The higher bandwidth provided by the SpaceWire protocol will allow for the adoption of advanced control schemes potentially based on multiple vision sensors and for the handling of sophisticated end-effectors that require fine control, such as science payloads or robotic hands. A feasibility assessment study has been performed recently. Preliminary system-level results have been presented at the Data Systems In Space (DASIA) Conference in 2015 [1]. Here, final and more detailed results regarding latency analysis are presented, including implications in terms of reliability and predictability of the behaviour of the LARAD control system.

SpaceWire has a technology development roadmap towards the support of Command & Control on board spacecraft [2]. The analysis performed in this study contributes and supports such roadmap by providing inputs from the space robotics domain.

In the remainder of the paper, the latency analysis is structured as follows. First, timing requirements of the LARAD OBC-JE interface (On-Board Computer – Joint Electronics) are established. Second, worst-case analysis related to the current CAN implementation is presented. Finally, a worst-case analysis for a SpaceWire implementation of the full control system is introduced and compared to the CAN version, covering the analysis of nominal and fault conditions, with and without the end-effector (EE).

II. Timing Requirements

The nominal rotation rate of the LARAD joint motors is around 9000 rpm. At this rate, full motor revolution happens at the frequency of 150 Hz, taking approximately 6.7 ms to complete. Design of the control system envisions that, in the worst case (i.e. when motion profile computation is performed by the OBC), motor control commands, consisting of position and rate demands for the PID controller, would have to be transmitted to the JE with 10 Hz frequency, i.e. at 100 ms intervals. Ten Hertz is also the frequency at which JE telemetry is acquired.

The above figures allow concluding that in order to achieve smooth motion, JE must receive and process the position/rate demand within 100ms after the previous one. In order to provide sufficient margin for the delays connected with processing, it may be concluded that the worst-case latency resulting purely from communication should be kept within one order of magnitude less that the deadlines discussed above, i.e. it should not exceed 10 ms.

III. Worst-case Latency Analysis – CAN

CAN has a bus architecture with a well-defined frame sizing and medium arbitration; the mechanism of prioritization is a key determinant for the communication delays. These features simplify worst-case latency analysis.

Table I shows the packets prioritisation in the current version of the LARAD Data Communication Definition between the OBC and Joint Electronics. Lower values take priority. "JE-X" designates a specific JE module (X can take values 0-5), "*" indicates broadcast.

TABLE I. PACKETS PRIORITISATION OF THE LARAD OBC-JE PROTOCOL

Message priority	Message name	Sender	Recipient
0	Emergency Stop	OBC	JE-X or *
1	Non-emergency Stop	OBC	JE-X or *
2	Position Control	OBC	JE-X
3	Torque Control	OBC	JE-X
4	Start Motion	OBC	JE-X
5	Brake Control	OBC	JE-X
6	Parameter Update	OBC	JE-X
7	Error Message	JE-X	JE-X
8	Telemetry	JE-X	JE-X

Considering message size, frame overhead due to the CAN standard and bit stuffing mechanism [3], the worst-case estimates shown in Table II are calculated for LARAD, assuming the bit rate of 1Mbps and extended frame format.

TABLE II. WORST-CASE SINGLE PACKET TRANSMISSION LATENCIES (CAN, EXTENDED FRAME FORMAT, 1 MBPS BIT RATE, 6-BIT INTER-FRAME DELAY)

Message name	Worst-case stuffing bits	Worst-case frame size [bits]	Worst-case single frame delay [s]	Worst-case total packet delay [s]
Emergency Stop	17	105	105.0E-6	105.0E-6
Non-emergency Stop	17	105	105.0E-6	105.0E-6
Position Control	25	153	153.0E-6	153.0E-6
Torque Control	22	134	134.0E-6	134.0E-6
Start Motion	19	115	115.0E-6	115.0E-6
Brake Control	19	115	115.0E-6	115.0E-6
Parameter Update	25	153	153.0E-6	630.0E-6
Error Message	20	124	124.0E-6	124.0E-6
Telemetry	25	153	153.0E-6	630.0E-6

The analysis of all possible combinations of messages with the features showed in Table II yields a worst-case sequence of messages potentially leading to the worst-case latencies for the messages transmitted over CAN bus given in Table III.

TABLE III. WORST-CASE OBC-JE TRANSMISSION LATENCIES (CAN, EXTENDED FRAME FORMAT, 1MBPS BIT RATE, 6-BIT INTER-FRAME DELAY)

Message name	Worst-case transmission delay [s]
Emergency Stop	105.0E-6
Non-emergency Stop	105.0E-6
Position Control	153.0E-6
Torque Control	134.0E-6
Start Motion	115.0E-6
Brake Control	115.0E-6
Parameter Update	630.0E-6
Error Message	7.0E-3
Telemetry	10.2E-3

Table II reveals that the worst-case latency upper bound for the Telemetry message meets the timing requirements discussed in Section II. However, values are close to the established acceptability threshold (10 ms). The future use of more sophisticated end effectors and control strategies involving additional sensors requires higher bandwidth, beyond the possibilities offered by CAN.

IV. WORST-CASE LATENCY ANALYSIS METHOD FOR SPACEWIRE

The SpaceWire protocol allows for the setup of networks with arbitrary topology and traffic. Therefore, calculating worst-case latencies in this kind of networks is not trivial. A pessimistic upper-bound method to calculate the actual worst-case latency values has been proposed [4], which inherently over-estimates the figures to a certain extent, similarly to what was done with the analysis for CAN presented in the previous section.

Unlike the current CAN-based control system, the LARAD SpaceWire control system will have to be able to host an exchangeable End Effector (EE). As a result, the resulting packet flows are summarised in Table IV.

TABLE IV. PACKET FLOWS IN THE LARAD CONTROL SYSTEM SPACEWIRE NETWORK

Packet Flow	Source	Destination	Packet size T_f
$f_{OBC \rightarrow JE\text{-}X}$	OBC	JE-*X* for *X* = 0..5	24
$f_{JE\text{-}X \rightarrow OBC}$	JE-*X* for *X* = 0..5	OBC	24
$f_{OBC \rightarrow EE}$	OBC	EE	$T_{OBC \rightarrow EE}$
$f_{EE \rightarrow OBC}$	EE	OBC	$T_{EE \rightarrow OBC}$

The End Effector is intended to be exchangeable. Therefore, the properties of the flows $f_{EE \rightarrow OBC}$ and $f_{OBC \rightarrow EE}$ cannot be fixed at this point. They will be treated as additional variables in the analysis. This leads to the following parameters on which worst-case network latencies depend:

- Spacewire network topology and packet routing;
- Link bit rate;
- Maximum sizes of the packets transmitted between the EE and the OBC in both directions.

A parametric analysis focusing on these three factors is presented in the next sections.

V. SPACEWIRE LATENCIES ANALYSIS – NO END EFFECTOR

In a configuration with no end effector, EE and JE traffic do not interact. This case provides a baseline for comparison with the CAN-based design (Section III), in which the EE is controlled via a dedicated bus. Furthermore, this will cover the

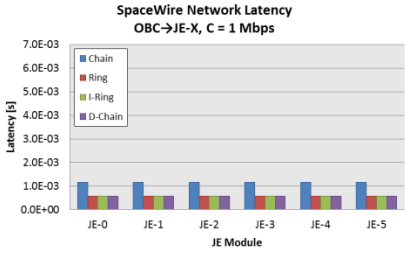
nominal scenarios for those network topologies which allow routing motor control flows independently of the End Effector data flows (i.e. doubly-linked chain and ring topologies); note that all the network topologies allowed in LARAD are discussed in details in a previous publication [1].

Assuming a link bit rate C of 1 Mbps, the method of Ferrandiz et al. yields, for the chain, ring, interleaved ring, and doubly-linked chain topologies, the worst-case latency upper bounds shown in Fig. 1.

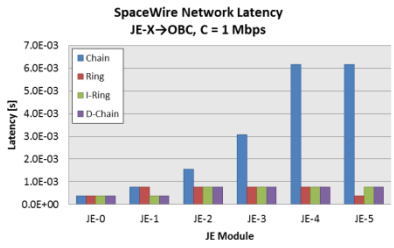
The method of Ferrandiz et al. [4] provides what can be argued to be very pessimistic latency figures. In order to get a better idea about the expected performance of the SpaceWire implementation of the LARAD control system, let us consider an optimistic scenario in which packets do not delay one another. In this case, network latency is simply the time necessary to transmit the longest packet between the furthest JE node and the OBC. This results in latency and corresponding maximum achievable control loop frequency figures shown in Table V.

Figure 1a shows the obtained upper-bounds on worst-case latencies for the flows $f_{OBC \rightarrow JE\text{-}X}$ (telecommand), and Fig. 1b for the flows $f_{JE\text{-}X \rightarrow OBC}$ (telemetry).

Note the considerable (one order of magnitude) disparity between the pessimistic and optimistic estimates obtained for the chain topology for the 1 Mbps link bandwidth. The behaviour of the network in practice will be somewhere between these figures.



(a)



(b)

Fig. 1. Worst-case SpaceWire network latencies without the EE data flows (C = 1 Mbps) for Joint Electronics telecommand (a) and telemetry (b)

TABLE V. OPTIMISTIC SPACEWIRE NETWORK LATENCIES WITHOUT THE EE DATA FLOWS FOR JOINT ELECTRONICS TELECOMMAND AND TELEMETRY

C [Mpbs]	1		10		200	
Topology	**Latency [s]**	**Freq. [kHz]**	**Latency [s]**	**Freq. [kHz]**	**Latency [s]**	**Freq. [kHz]**
Chain	195.00 E-06	5.13	22.20E-06	45.05	3.96E-06	252.53
Ring / I-Ring / D-Chain	193.50 E-06	5.17	20.70E-06	48.31	2.46E-06	406.50

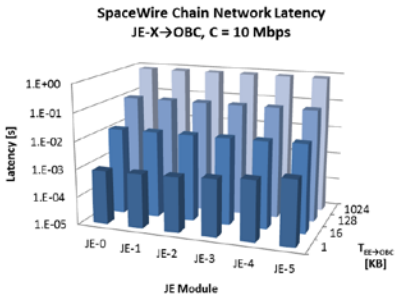
Calculations presented so far in this section assume that the End Effector is either not present, it does not communicate with the OBC during the arm movement, or its data flows are routed separately from the JE data flows. The latter is achievable in the doubly-linked chain and ring topologies. In this case, these become equivalent to the chain topology without the EE traffic and thus are expected to offer the same performance. In the case of the remaining SpaceWire network topology options (chain and interleaved ring), for the analysis to be more representative, the interaction between the motor control and End Effector traffic needs to be taken into account. The same applies for the doubly-linked chain and ring topologies when some of the links are lost due to failures. We investigate these scenarios in the next sections.

VI. SPACEWIRE LATENCIES ANALYSIS – CHAIN AND INTERLEAVED RING TOPOLOGIES WITH EE TRAFFIC

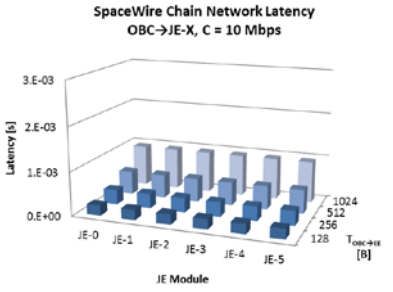
The LARAD End Effector is intended to be exchangeable. Therefore, a parametric analysis is proposed here, assuming different maximum sizes of the End Effector traffic packets (for both telecommand and telemetry).

Assuming that the ranges of realistic telecommand sizes is covered by the values of $T_{OBC \rightarrow EE}$ of 128, 256, 512, and 1024 bytes, and of realistic telemetry sizes by the values of $T_{EE \rightarrow OBC}$ equal to 1, 16, 128, and 1024 KB, the upper bounds on worst-case latencies of the OBC-JE communication in the chain network with link speed of 10 Mbps are as shown in Fig. 2.

The overall conclusion from the latency analysis for the network with chain topology is that, should it be adopted, careful attention would have to be paid to the properties of the End Effector protocols: End Effector communication shall not disrupt the operation of the LARAD control system.



(a)

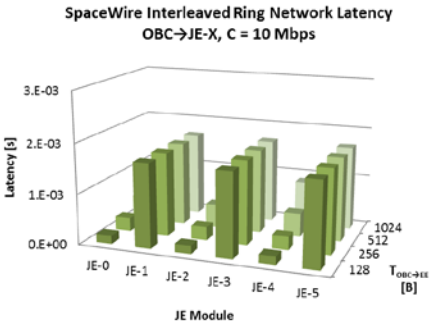


(b)

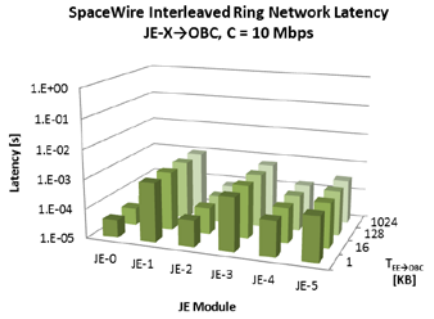
Fig. 2. Worst-case SpaceWire latencies in a network with chain topology (C = 10 Mbps) for JE telecommand (a) and telemetry (b)

More optimistic conclusions can be drawn in the case of the Interleaved Ring network topology. Although End Effector and Joint Electronics data flows cannot be routed completely independently in this topology, it is possible to set up packet routing in such a way that End Effector telemetry does not interfere with any other data flow [1].

The upper bounds on worst-case latencies for a network with such routing, calculated under the same conditions as for the chain topology, are shown in Fig. 3. EE telecommand data flows introduce increased latencies on the data flows of the JE modules which they share communication links with. However, the interference is at acceptable levels for all considered packet sizes. Maximum obtained latency bounds are 0.992 ms for JE telemetry and 1.68 ms for JE telecommand (both for JE-1). Note that in the case of JE telemetry, latency values do not depend on the EE telemetry maximum packet size, as the flows do not interact. Increased latencies in the case of JE-1, JE-3, and JE-5 are caused by the fact these telemetry flows interact with their own telecommand flows, as they are all transmitted in the same direction along the ring.



(a)



(b)

Fig. 3. Worst-case SpaceWire latencies in a network with interleaved ring topology (C = 10 Mbps) for JE telecommand (a) and telemetry (b)

The qualitative conclusion from the analysis in the presence of the EE is that EE telemetry traffic, due to its large packet sizes, can be very disruptive in terms of latency performance to the data flows it shares communication links with. In order to guarantee that motor control communication meets the assumed deadlines at the link speed of 10 Mbps, the maximum allowable size of the EE Telemetry packet in the case of the chain topology is only around 16 KB. The case of the interleaved ring topology demonstrates that if EE telemetry traffic can be isolated, much more optimistic latency estimates are obtained. Even if sophisticated routing set-up is needed, figures indicate it is worth the effort.

VII. SpaceWire latencies analysis – Contingency scenarios

The distinct advantage of the doubly-linked chain, ring, and interleaved ring topologies in comparison to the chain topology is the resistance of the former three to link failures. For the first two topologies, a failure of any of the SpaceWire links (except for the internal links between JEs and their built-in SpW Switches) means that EE messages cannot be routed independently from the motor control messages any more. The extent to which link loss affects worst-case latency estimates for each of these three topologies is assessed in this section.

Multiple different points of failure are possible for doubly-linked chain, ring, and interleaved ring topologies. Here the analysis focuses on the representative cases shown in Fig. 4.

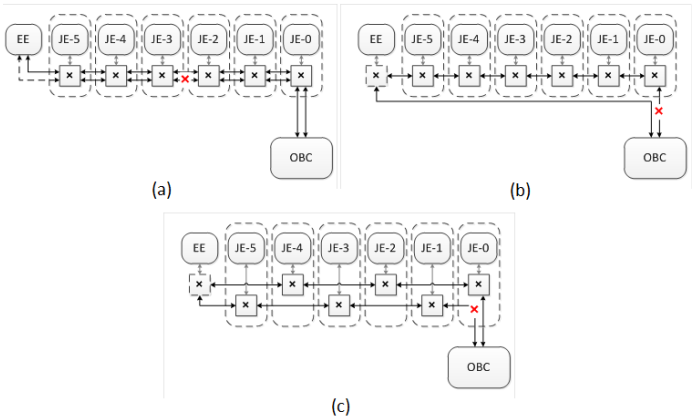
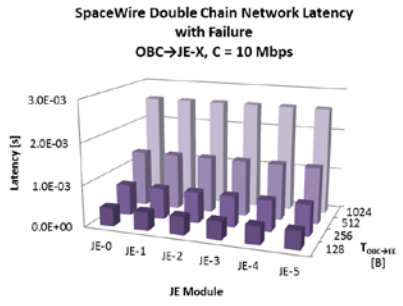


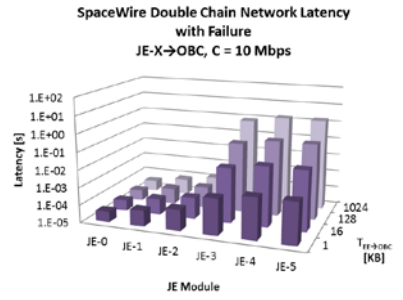
Fig. 4. Example failure points (red crosses) chosen for the doubly-linked chain topology (a), ring topology (b) and interleaved ring topology (c)

For telecommands, the estimated latencies are all well within the assumed acceptable range (10 ms). The largest figures have been obtained for the doubly-linked chain topology – about 2.6 ms for $T_{OBC \rightarrow EE}$ of 1 KB, and about 1.4 ms for $T_{OBC \rightarrow EE}$ of 512 B; in all other cases the estimated worst-case latencies are below 1 ms. The reason behind increased estimates for the doubly-linked chain topology is that in this case the interaction between the JE and EE data flows may be indirect, and therefore the delay caused by EE packets may be inflicted multiple times (e.g. flow $f_{OBC \rightarrow JE\text{-}0}$ is potentially delayed by flows $f_{OBC \rightarrow JE\text{-}3}$, $f_{OBC \rightarrow JE\text{-}4}$, and $f_{OBC \rightarrow JE\text{-}5}$, each of which can be delayed on the link between JE-2 and JE-3 routers by packets belonging to $f_{OBC \rightarrow EE}$). In situations depicted in Fig. 4b and in Fig. 4c, according to the calculation method [4] adopted, the delay caused by the EE packets may be inflicted only once, in the OBC node, in which all interacting packet flows originate. As the result, figures obtained for telecommands sent over ring and interleaved ring topologies are identical. As may be expected, they are also equivalent to the performance of the simple chain topology without failures.

Upper-bounds on the worst-case network latencies calculated for the three scenarios for both telecommands and telemetries are shown in Fig. 5, Fig. 6 and Fig. 7.

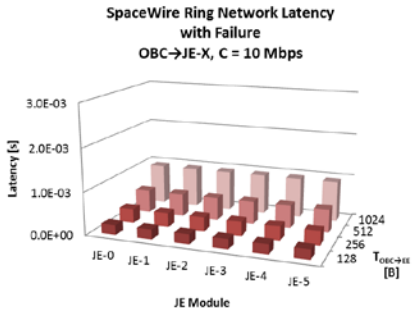


(a)

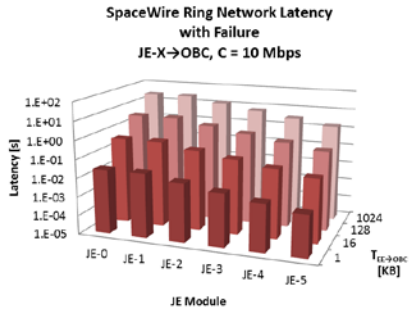


(b)

Fig. 5. Worst-case SpaceWire latencies for the JE telecommand (a) and telemetry (b) data flows in the doubly-linked chain network topologies with failures as per Fig. 4a



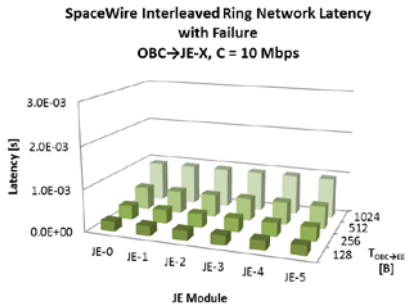
(a)



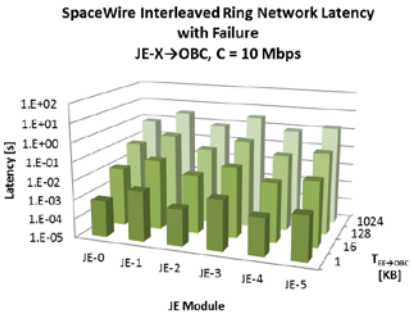
(b)

Fig. 6. Worst-case SpaceWire latencies for the JE telecommand (a) and telemetry (b) data flows in the ring network topologies with failures as per Fig. 4b

The general conclusion from the analysis of the worst-case latencies in the fail-over scenarios is that although the considered network topologies offer resistance to failures in the sense of network connectivity, the resulting jitter in communication timings may cause it to be challenging to guarantee reliable operation of the LARAD control system in parallel with the End Effector payload.



(a)



(b)

Fig. 7. Worst-case SpaceWire latencies for the JE telecommand (a) and telemetry (b) data flows in the interleaved ring network topologies with failures as per Fig. 4c

VIII. Conclusions

A system-level evaluation of a new SpaceWire version of the LARAD control system has revealed a number of benefits compared to the current CAN-based version [1]. The following SpaceWire network topologies have been taken in consideration for LARAD: star, chain, doubly-linked chain, ring, interleaved ring.

The latency analysis results presented in this paper contribute to the general conclusion that the interleaved ring topology provides the best trade-off in terms of performance and reliability of the overall control system.

The SpaceWire-based version of the LARAD control system was analysed with asynchronous communication because of assumptions on the functioning of the End Effector; this has the by-product of high jitter. Further analysis should consider the use of SpW-D 2.0 which will provide a mix of synchronous and asynchronous communication services, hence reducing jitter to very low values for critical communications (LARAD control).

References

[1] M. Rucinski, A. Coates, G. Montano, E. Allouis and D. Jameux, “SpaceWire-based Control System Architecture for the Lightweight Advanded Robotic Arm Demonstrator (LARAD),” in *DAta Systems In Space (DASIA)*, Barcelona, Spain, 2015.

[2] D. Jameux, “SpaceWire for Command & Control,” ESA, Noordwijk, The Netherlands, 2015.

[3] Robert Bosch GmbH, “CAN Specification,” 1991.

[4] T. Ferrandiz, F. Fabrice and C. Fraboul, “Worst-case end-to-end delays evaluation for SpaceWire networks,” *Discrete Event Dynamic Systems* , vol. 21, no. 3, pp. 339-357, 2011.

SpaceFibre Port IP Core (GRSPFI)

SpaceFibre, Poster Paper

Felix Siegle, Sandi Habinc
Cobham Gaisler AB
Gothenburg, Sweden
[felix | sandi] @gaisler.com

Johannes Both
European Space Agency
Noordwijk, The Netherlands
johannes.both@esa.int

***Abstract*—Cobham Gaisler presents the SpaceFibre Port IP Core implementation GRSPFI. A fully validated VHDL implementation is readily available.**

***Index Terms*— ASIC, FPGA, Networking, SpaceFibre**

I. Introduction

SpaceFibre is a new high-speed serial data link specifically designed for spaceflight applications that incorporates several Quality-of-Service (QoS) techniques. Independent communication channels can be combined into a single network stream by means of virtual channels. The virtual channels are multiplexed based on reserved bandwidth, priorities, time-slots, or a combination of these mechanisms. Integrated Fault Detection, Isolation and Recovery (FDIR) support guarantees fault-free communication. Comparable to other modern intercommunication architectures like Serial RapidIO, Serial ATA or PCI Express, SpaceFibre communicates over a Serialiser/Deserialiser (SerDes) device and can therefore reach throughput rates of several Gigabits Per Second (Gbps). However, due to its native support of the SpaceWire packet format, its small area overhead and its high performance, SpaceFibre is particularly well suited for future high-speed on-board communication.

Cobham Gaisler closely follows the standardization efforts of the European Cooperation for Space Standardization (ECSS), which will soon publish the SpaceFibre Specification E-ST-50-11C. Already now, Cobham Gaisler can provide a draft single-lane SpaceFibre IP core that can easily be implemented on modern FPGA devices like Xilinx Virtex-5 or Microsemi RTG4.

II. SpaceFibre

SpaceFibre is a high-speed serial link mainly designed for payload data processing applications on board spacecraft. Like many other modern network architectures, SpaceFibre utilises a SerDes circuit at its physical layer, allowing data rates of 2 Gbps and more. The SerDes can either be part of the chip design or a standalone device can be used.

Interfacing a SpaceFibre port from the user application is simple as it closely follows the procedure known from SpaceWire. A SpaceFibre port has one or more pairs of transmit and receive buffers, referred to as virtual channels, and each virtual channel acts like a single SpaceWire interface,

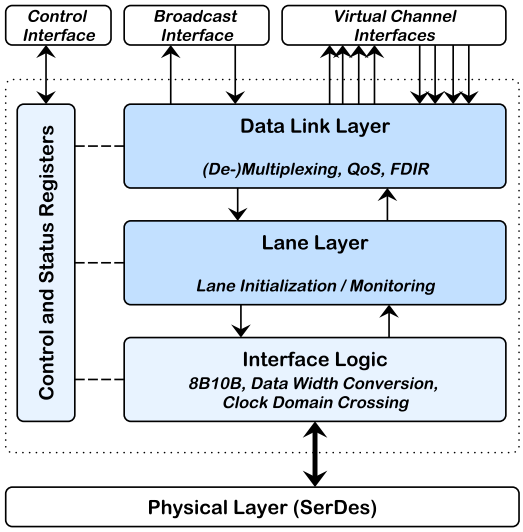


Fig. 1. Simplified block diagram of the GRSPFI SpaceFibre Port IP Core.

i.e. several SpaceWire network streams can be multiplexed into one SpaceFibre network stream. The multiplexer is called medium access controller and is choosing the active virtual channel according to a number of Quality-of-Service rules.

Data is always transferred in frames with a size of 256 bytes or less. While such a data frame is passed to the physical link, it is also stored in an error recovery buffer. It remains in this buffer until the destination node acknowledges the correct reception of the frame, which is detected by checking a CRC checksum at the end of the frame. However, if the destination node sends a negative-acknowledgement (NACK) word instead, the frame is re-transmitted from the error-recovery buffer.

Aside from data frames, SpaceFibre also supports broadcast frames, which are multi-purpose high-priority messages. These messages are comparable to SpaceWire time-codes but in addition to a simple sequence number they also comprise a data payload of 8 bytes. Broadcast frames are stored in the error recovery buffer just like the data frames, i.e. they are automatically retransmitted after a link error.

On the receive side, incoming data from the physical link is processed continuously, i.e. one 32-bit word is processed every clock cycle. To avoid buffer overruns in the virtual channel receive buffers, the communication between a virtual channel transmit buffer in the local node and the virtual channel receive

buffer in the destination node is flow-controlled by means of Flow Control Token (FCT) words. Just like the data and broadcast frames, the FCT words are stored in the error recovery buffer and are therefore retransmitted in case of errors.

III. GRSPFI – SpaceFibre Port IP Core

A simplified block diagram of the SpaceFibre IP core can be seen in Figure 1. As described in [1], the SpaceFibre IP port comprises a data link layer and lane layer. Internally, the data link layer is further divided into the so-called broadcast layer, virtual channel layer and retry layer, which are responsible for the transmission and reception of broadcast frames, for the transmission and reception of data frames and for the error recovery mechanism, respectively.

A. Hard Configuration Options

Instantiating the GRSPFI IP Core is straight-forward. The following configurations options are available at compile time through VHDL generics:

1) Virtual Channels

- The number of virtual channels.
- The depth of the receive buffers.
- The depth of the transmit buffers.
- Width of the data bandwidth credit counter.
- Bandwidth idle time limit in clock cycles.

2) Broadcast Channel

- Width of the broadcast bandwidth credit counter.
- Minimum bandwidth credit threshold limit.

3) Error-Recovery

- Depth of the data error recovery buffer.
- Depth of the FCT error recovery buffer.
- Depth of the broadcast error recovery buffer.

4) SerDes Interface

- With internal 8B/10B coding: 20-bit or 40-bit interface, without internal 8B/10B coding: 18-bit or 36-bit interface.
- Enable/disable internal 8B/10B coding.
- Use a separate transmission clock: This feature is mandatory if the 18-/20-bit SerDes interface is used and optional in case of the 36-/40-bit interface.

5) Technology

- Use asynchronous or synchronous reset.
- Memory technology: enables the automatic instantiation of technology-dependent internal RAMs, including fault-tolerant versions, for different ASIC and FPGA families.

B. Soft Configuration Options

The following options can be set during runtime:

- Lane Start & Autostart.
- Internal loopback.
- Enable/disable data scrambler.
- Expected broadcast bandwidth value.
- An expected virtual channel bandwidth value for each virtual channel.
- A timeslot vector for each virtual channel.
- A priority value for each virtual channel.

C. Status Registers and Flags

The GRSPFI IP Core implements all status registers and flags as required by [1]. They indicate the current state of the lane (e.g. lane state and RXERR word counter) and data link layer (e.g. error-recovery retry count, sequence errors, CRC16 errors, virtual channel bandwidth over- and underuse).

D. Interfacing the SerDes

Connecting the GRSPFI IP Core to a SerDes is done through an interface logic that allows various clocking and data width schemes, supporting a wide range of available on-chip and off-chip SerDes circuits. On the transmit side, it comprises the optional 8B/10B encoder and an optional asynchronous FIFO used for data width conversion and clock domain crossing. On the receive side, it comprises the optional 8B/10B decoder, the optional data path with conversion logic, the word synchronisation logic and the elastic buffer.

Example 1: A Xilinx RocketIO GTX transceiver [2] with its own 8B/10B coding logic can be interfaced through a 36-bit interface. Then, only one transmission clock is needed for both the GTX transceiver and the GRSPFI IP Core.

Example 2: The SerDes of a Microsemi RTG4 device [3] must be interfaced through its 20-bit wide EPCS interface and does not include its own 8B/10B coding logic. In this case, the GRSPFI IP Core can be configured to implement the 8B/10B encoder and decoder and the asynchronous FIFO on the transmit side. Then, two transmission clocks are required, one for the SerDes and one derived clock for the GRSPFI IP Core at half the frequency. For instance, to achieve an effective link speed of 2.5 Gbps, the SerDes would need to be clocked at 125 MHz and the GRSPFI IP Core at 62.5 MHz.

IV. Verification and Validation

The IP core is fully verified by means of a VHDL testbench system and validated in hardware.

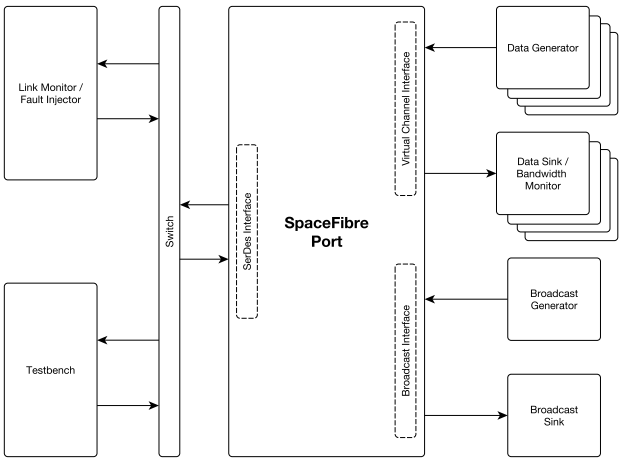


Fig. 2. Block diagram of the GRSPFI SpaceFibre Port IP Core testbench.

A. *Verification*

The default testbench setup comprises a SpaceFibre port with four virtual channels as depicted in Figure 2. Four data generator and data sink processes are connected to the virtual channel interface as well as one broadcast generator and broadcast sink process to the broadcast interface. The signals to the SerDes can either be looped back to the SpaceFibre port through a link monitor process or directly fed into the testbench process, allowing two basic operation modes:

- The SpaceFibre port is operated in (external) loopback mode, i.e. data and broadcasts transmitted by one of the generator processes arrive in the corresponding sink process and the link initialisation handshake is done automatically between the transmit and receive side of the SpaceFibre port.
- The SerDes signals are directly stimulated, i.e. data and broadcasts transmitted by one of the generator processes arrive in the testbench process and the link initialisation handshake is managed by the testbench process.

The first operation mode is well suited for testing most Quality-of-Service features of the port whereas the second operation mode is particularly useful for driving the port into states, which are rarely hit during normal operation.

The testbench executes 45 tests altogether, including tests covering all aspects of the lane layer, the virtual channel data communication and flow control, the virtual channel QoS mechanisms (bandwidth allocation, priorities, timeslots, babbling idiot protection), the reception and transmission of broadcasts, the correct behaviour of the error-recovery mechanisms and the data scrambler and de-scrambler.

The IP core was verified according to guidelines of the European Space Agency (ESA). Full code coverage was achieved, i.e. 100% statement coverage, 100% branch coverage and 100% FEC condition terms coverage.

B. *Validation*

The SpaceFibre IP has been successfully verified in a hardware testbench as depicted in the block diagram in Figure 3. The IP is configured to have four virtual channels and one data test block is connected to each of these channels. A data test block is able to send and receive SpaceWire packets through its virtual channel with maximum bandwidth. On the receive side, incoming data is checked for correctness by means of a sequence number as well as detectors for EOP and EEP characters. In addition, a timer is running throughout the reception of data, allowing the calculation of the average data throughput rate. Similarly, a broadcast test block is connected to the broadcast interface of the SpaceFibre IP that is able to send and receive broadcast frames at a configurable frequency. Again, received broadcasts are checked for correctness and a timer allows the calculation of the throughput rate. In addition, a counter keeps track of the number of broadcasts that were received with a Late flag set to 1. The configuration and status registers of the data and broadcast test blocks are accessible through an APB bus interface.

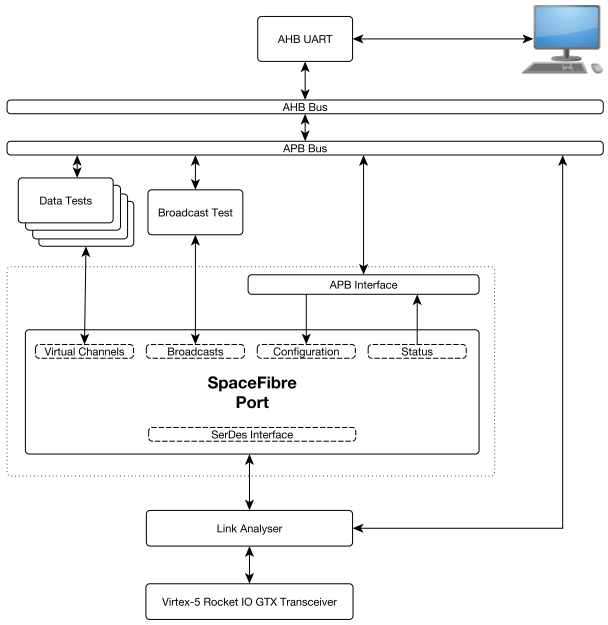


Fig. 3. Block diagram of the hardware validation setup.

A link analyser is placed between the SpaceFibre IP and the RocketIO GTX transceiver, allowing the monitoring of incoming and outgoing traffic. The link analyser block can trigger on specific SpaceFibre control words or specific word content. A trace buffer is then filled with 8192 values, which can later be read out through an APB interface.

Communication with the hardware testbench is done through a UART block that is connected to the APB bus. On the host PC, GRMON [4] is responsible for setting and reading the configuration and status registers. Within GRMON, a TCP server listens for commands. A graphical user interface has been

designed that allows a quick and easy validation of the hardware. A brief overview of its functionality is given in the next sections.

1) *Test Software*

a) *Port Configuration*

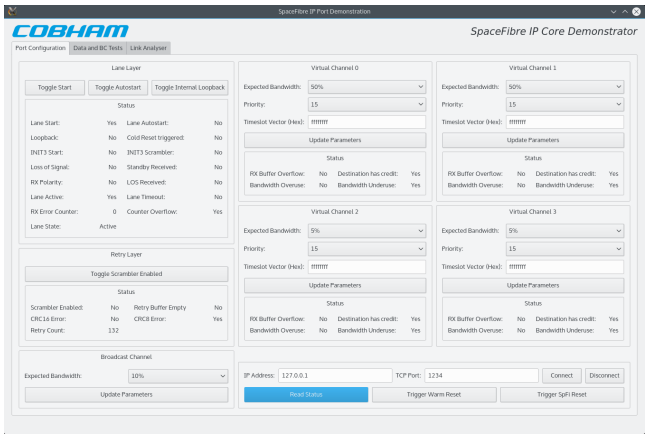


Fig. 4. Screenshot of the GUI: Port configuration.

The following functions are available on this first tab, see Figure 4:

- Lane Layer: The 'Start', 'Autostart' and 'Internal Loopback' control flags can be toggled. All status flags and values are displayed.
- Retry Layer: The 'Scrambler Enabled' flag can be toggled. All status flags and values are displayed.
- Broadcast Channel: The expected bandwidth value can be chosen (1% to 95%).
- Virtual Channels: The expected bandwidth value can be chosen (1% to 95%), the priority level can be chosen (0 to 15) and a timeslot vector can be defined as hex value.
- A warm reset and SpFi system reset can be triggered.

b) *Data and Broadcast Tests*

The following functions are available on the second tab, see Figure 5:

- Data Tests: Up to 4 SpaceWire addresses, the packet length as well as the number of packets can be defined. The transmit and receive side can be enabled separately and an auto-repeat function allows the continuous generation of packets. Important status information is shown, including the average throughput rate for the last block of packets.
- Broadcast Test: The number of broadcasts, the broadcast channel as well as the transmission delay between each broadcast can be defined. Similarly to the data test blocks, the transmit and receive side can be enabled separately and an auto-repeat function allows the continuous generation of broadcasts. Important status information is shown, including the average throughput rate for the last block of broadcasts.

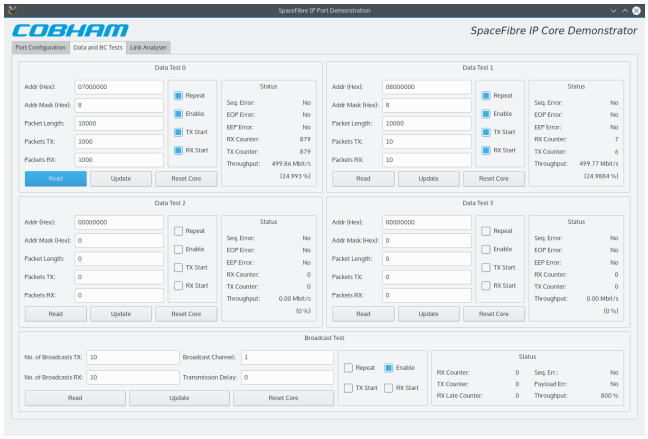


Fig. 5. Screenshot of the GUI: Data and broadcast tests.

c) *Link Analyser*

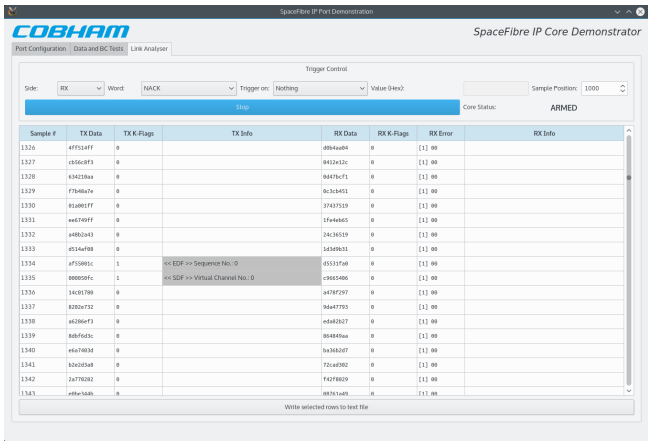


Fig. 6. Screenshot of the GUI: Link analyser.

On this tab, see Figure 6, link traffic to/from the SerDes can be analysed. It is possible to trigger on either the receive or transmit side on the following words: Any, Data/Broadcast Payload, SKIP, IDLE, INIT1, INIT2, INIT3, STANDBY, LOS, ACK, NACK, FULL, RETRY, SDF, EDF, SBF, EBF, FCT, SIF and RXERR. Depending on the word, further trigger conditions can be added. For instance, the SDF word allows the triggering on a specific virtual channel number or the EBF word allows the triggering on a specific sequence number or late flag. In addition, a sample trigger point can be defined. Once the core is armed, the software is waiting for the trigger condition to become true. Then, the 8192 values are transferred to the host PC and displayed in a table. The table displays the raw data word, the corresponding K-flags as well as information about control words. On the receive side, the RX error flags and the byte alignment flags of the SerDes are displayed as well. Multiple rows of the table can be selected and exported as a text file.

2) Test Setup, Cases and Results

The testbench uses a RocketIO GTX transceiver tile available on the Virtex-5 device, which is connected to Serial ATA connectors on the Xilinx ML510 development board. The internal data width of the SerDes is 20 bits with a target line frequency of 2.5 Gbps. The internal 8B/10B encoder/decoder is enabled and the interface data width is 32 bits. The reference clock is 100 MHz and provided by the system clock of the ML510 board.

The GTX transceiver's PLL provides a 125 MHz clock (TXOUTCLK), which is used to feed both the TXUSRCLK and RXUSRCLK. For the 32-bit wide interface and the SpaceFibre IP core a 62.5 MHz clock is needed, however, and therefore TXOUTCLK is first fed into a DCM that generates this phase-aligned divided clock.

Standard settings for pre-emphasis and differential swing are used, however, RX equalization is enabled. The receive side is terminated with a termination voltage of 2/3 * VTTRX. The comma detection is set up to detect and align on positive and negative K28.5 commas. Clock correction for the elastic buffer is enabled and set up to use the SKIP word as clock correction sequence.

Near-end PMA loopback was enabled in the RocketIO Transceiver, i.e. the transmitted data was serialised, looped back, de-serialised and fed back to the receive side of the SpaceFibre IP core. This configuration allowed in-depth testing of all cases listed in Table I. They cover all aspects of SpaceFibre and were partly conducted over the course of days or even weeks to ensure data integrity and that the logic is driven in all possible states.

TABLE I. HARDWARE VALIDATION TEST CASES

Test Case Description	**Result**
Link initialisation handshake test (link start and auto start mode).	Passed
Transmission and reception of data frames.	Passed
Data frames interrupted by link resets are resent correctly.	Passed
Transmission and reception of broadcast frames.	Passed
Broadcast frames interrupted by link resets are resent correctly.	Passed
Scrambled data frames transmitted/received correctly.	Passed
Broadcast transmissions do not exceed maximum allowed bandwidth value.	Passed
Priority mechanism for virtual channels works as expected.	Passed
Timeslot mechanism for virtual channels works as expected.	Passed
Bandwidth limitation of virtual channels works as expected.	Passed
Internal loopback mode works as expected.	Passed

V. IMPLEMENTATION RESULTS

Example post-P&R results for Virtex-5 FX130 are given in Table 2. A typical implementation with 4 virtual channels (1024-words deep transmit and receive buffers) was chosen.

TABLE II. IMPLEMENTATION RESULTS ON VIRTEX-5 FX130

Max. Throughput Rate:	> 6.25 Gbps (f = 156.25 MHz)
Slice LUTs:	4318/81920 (5%)
Slice Flip-flops:	1892/81920 (2%)
Block RAMs:	11/298 (3%)

As can be seen from the results, the pipelined structure of the GRSPFI IP Core allows high throughput rates while keeping the resource utilisation low. Similar results can be expected for Microsemi's RTG4 device and preliminary experiments showed that small configurations (e.g. with 2 virtual channels) can also be implemented on older devices like Microsemi's RTAX2000.

VI. CONCLUSIONS

Cobham Gaisler offers with the GRSPFI SpaceFibre Port IP core a fast and easy to implement single-lane implementation of SpaceFibre with low area and power overhead. It is fully verified in a complex VHDL testbench environment and validated by means of a FPGA prototype system that also includes a versatile test and debug software.

REFERENCES

[1] Space Technology Center/University of Dundee, "SpaceFibre Specification Draft H1", Aug. 2015.

[2] Xilinx, "Virtex-5 FPGA RocketIO GTX Transceiver. User Guide UG198". Xilinx, Oct. 30, 2009. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug198.pdf.

[3] Microsemi, "RTG4 FPGA High-Speed Serial Interfaces. User Guide UG0567". [Online]. Available: http://www.microsemi.com/document-portal/doc_view/134409-ug0567-rtg4-fpga-high-speed-serial-interfaces-user-guide.

[4] Cobham Gaisler, "GRMON Website". [Online]. Available: http://www.gaisler.com/index.php/products/debug-tools/grmon.

Synchronization of Distributed Interrupts Delivery in Aerospace Onboard Networks

Network and protocols, Short Paper

Liudmila Koblyakova, Elena Suvorova, Yuriy Sheynin
Institute of High-Performance Computer and Network Technologies
St. Petersburg State University of Aerospace Instrumentation
St. Petersburg, Russian Federation
liudmila.koblyakova@guap.ru, suvorova@aanet.ru, sheynin@aanet.ru

***Abstract* - SpaceWire is integrated networking technology where the data packets and control traffic in the form of hard real-time signals are transmitted throw the same links of communication network. Last decades for hard real-time signals transmission in onboard networks the specialized buses like CAN and MIL-STD 155b and sideband signals were used. All buses have the property that the signal from the source node to all other nodes propagates simultaneously, so there is no jitter for signal delivery. When the industry moves to a common communication medium for control and data traffic transmission through network with routing switches and point-to-point connections, there is a problem of synchronization hard real-time signals delivery from the source to all other nodes because of different distance (number of switches and link speeds in a path) between source and receiver.**

There are many different algorithms and methods for the system time synchronization. In this paper we consider the questions of delivery synchronization of distributed interrupts which intended to control and inform the devices about critical system events in hard real-time.

***Index Terms*—hard real-time signaling, Distributed Interrupts, synchronization**

INTRODUCTION

SpaceWire standard has developed specially for aerospace applications, [1]. For every onboard system the important task is the control signal transmission in hard real-time.

The SpaceWire standard for hard real-time signal transmission uses the mechanism of time-codes transmission and the distributed interrupts mechanism. The distributed interrupts mechanism can work in two modes: with or without acknowledges. They can be used for:

- for control signal transmission in hard real-time (distributed interrupts in mode with acknowledges);
- for notification signals transmission in hard real-time (distributed interrupts in mode without acknowledges);
- for synchronization of distributed actions (distributed interrupts in mode without acknowledges).

In this paper we consider the use case of distributed interrupts mechanism for distributed actions synchronization.

FORMULATION OF THE DISTRIBUTED ACTIONS SYNCHRONIZATION TASK

Assume that it is necessary to synchronize the certain actions in the group of devices. We use the distributed interrupts in the mode without acknowledge. The source of the interrupt is the one pre-known node, which sends the distributed interrupt when:

- particular event appears and the time between events can be different, or,
- by timer, and the time between events is the same.

The recipients of these interrupts are pre-defined group of the devices in which some distributed action should be synchronized. The devices of this group could be concentrated at one part of the subnetwork/region, or could be distributed between different parts of network. The propagation time of the distributed interrupt code from the source to all recipients might be different, even though small, therefore there is the task of distributed interrupts delivery synchronization from the known source to the group of recipients which are known in advance, at system design stage. This task requires that jitter of interrupt code propagation time from the source to all recipients should be minimized.

The paper [2] describes the time characteristics of distributed interrupts in general and was basically aimed for estimation of maximum propagation time and timeouts values. Let consider the interrupt-code propagation time from the point of view of our problem.

The maximum propagation time of an interrupt code between most distanced nodes by the shortest path is defined in the paper [3]:

$$T_{\max} = L_{Queue} \cdot T_{cc} + (D-1) \cdot (T_{wtc} + T_{bit}(N_{cc}-1)) + D \cdot T_{cc}, \quad (1)$$

where L_{Queue} – the worst case of queue length with interrupts/acknowledges codes, D – the number of edges (links) in the shortest path between the most distanced nodes, T_{wtc} – interrupt-code propagation time through the router without taking into account the latency of previous codes transmission, T_{cc} – control code propagation time through the link, T_{bit} – propagation time of one bit through the link. In our task we are interested in the mean interrupt-code propagation time.

The Interrupt-code from the known source node to each of the known handler nodes propagates through the shortest path. Let P – a set of paths from the source node to the n handler nodes:

$$P = \{P_1, P_2, \ldots, P_n\} \quad (2)$$

Every path P has a length L_P of edges (links):

$$L_P = \{L_{P1}, L_{P2}, \ldots, L_{Pn}\} \quad (3)$$

The mean propagation time of the interrupt-code through the L_{Pk} path we could write as (4):

$$T_{L_{P_k}} = \sum_{i=1}^{L_P} \frac{N_{CC}}{S_i} + \sum_{i=1}^{L_P-1} (T_{wtc_i} + \frac{N_{Buf_i}}{S_i}), \quad (4)$$

where the first sum defines the transmission time through the links and S_i - is a speed in i-th links of L_P path, the second sum defines the interrupt-code propagation time through the routers in the path, where the first summand $Twtc_i$ defines interrupt-code propagation time without taking into account the waiting time for transmission of previous codes in every router, the second summand defines the transmission time of symbols which are the previous to the interrupt-code, where – N_{Bufi} the number of bit waiting for transmission in buffer of output port. For every port we get its value of mean propagation time T_{Lp} (5):

$$T_{L_P} = \{T_{LP_1}, T_{LP_2}, \ldots, T_{LP_n}\} \quad (5)$$

From the equation (4) is seen that jitter depends on the number of links in the path, the speeds in these links, and if the network contains the routers of different versions the T_{wtc} parameter can be different too. Also the value N_{Bufi} at each interrupt-code transmission at every router can be different. It is obvious that the task of interrupt-codes delivery synchronization is dependent on the specific structure of network, their parameters and interposition of the interrupt source and handlers (nodes which are necessary to synchronize). Therefore it is hardly ever possible to find the universal solution of this problem. Further we consider the different variants of solving this task.

SYNCHRONIZATION OF INTERRUPTS CODE DELIVERY IN A STATIC SYSTEMS

Under a static system we mean a system with known in advance topology, links speed, interposition of the interrupt source and all its handlers – group of nodes, which it is necessary to synchronize some actions in. For such a system at the stage of its design it is possible to determine theoretically the set of paths and their length. By the equation (4) it can be estimated the value of interrupt-code propagation time from the source to every handler, because links speed and the interrupt-code are known and the value of T_{wtc} is defined by the device manufacturer. It remains to determine the values N_{Buf} for every output port of every router from the set of paths. These values depend on the flow intensity of time-codes and distributed interrupts of other types on the paths of propagation interrupt-codes for synchronization and its value can be calculated accurate. Therefore these values can be chosen by the system designer only approximately, based on presumptive control code flows in a particular system.

Then we should calculate all values of the set T_{Lp} (5). Choose the maximum interrupt code propagation time from the set T_{LpMAX}. Further, based on this maximum value it is necessary to calculate the correction values T_{Corr} for all other recipient nodes.

$$T_{Corr} = \{T_{LP_{MAX}} - T_{LP_1}, T_{LP_{MAX}} - T_{LP_2}, \ldots, T_{LP_{MAX}} - T_{LP_n}\} \quad (6)$$

The corresponding value of T_{Corr} should be written in every recipient node. Then, when the interrupt code for synchronization is received, the node should wait during a time T_{Corr} before starting the action T_{Corr}, (7):

$$T_{Act} = T_{Receive} + T_{Corr} \quad (7)$$

SYNCHRONIZATION OF INTERRUPTS CODE DELIVERY IN DYNAMIC SYSTEMS

Under a dynamic system we mean a system in which during the operation devices or channels can be connected or disconnected, can be changed the link speeds and as consequence of this the traffic flows may be redistributed. Interrupt-codes are broadcasted from the source to all other nodes, so if occurred any changes, under which the network remains connected, the interrupt-codes would be delivered to all nodes in the network, but the paths of interrupts-code propagation could be changed in dependence with network state in transmission moment. That is why we can not calculate corrections before restarting the network and then use it during network operation. The variants of solving the interrupts-code delivery synchronization task in dynamical systems can be different, we consider them further.

The theoretical method of interrupt-code delivery synchronization

In current on-board networks methods of Plug and Play are actively developing and applying , which allow defining automatically network structure, including all connections and links speed, to configure and monitor the network state during the operation, [4, 5]. If the Plug and Play methods are used in the network, than the special node (or several nodes) – network manager, gets and contains all information about changes of link speed, in including/excluding devices and channels. In this case calculation of correction values could be done by a separate application on a node, which has the access to Plug and Play manager's data.

The interrupt-code source and handlers are known in advance; the shortest codes propagation paths and corrections for every recipient are calculated before network starting or during initialization. Further, when any changes in the network occur, the network manager checks whether these changes relate to the considered interrupt-code propagation paths. Next, for all paths where have been changes, the

application calculates new values of T_{Lp}, and check if the T_{LpMAX} value has changed or not. If T_{LpMAX} value has not changed, then it is required to recalculate the correction value T_{Corr} only for nodes, which signal propagation path has changed. If the maximum value T_{LpMAX} has changed, it is required to recalculate the correction values T_{Corr} for all recipient nodes. To provide possibility for changing correction values T_{Corr} during the network operation, they should be stored in registers or some memory area, which allow access for writing by the RAMP protocol, [6].

This method does not required large overheads for the traffic transmission of statistics, and require only existence of Plug and Play methods which monitor the changes in a system and some processing resources in one of the nodes, for example in the source node of in the manager-node of Plug and Play.

The statistical methods of interrupt-code delivery synchronization

In the statistical method, in contrast to the theoretical, the interrupt-code delivery time to every recipient node should be estimated by experiments. Statistics should be collected based on the which the correction values could be calculated. For the statistic calculation it is required to measure the interrupt-code propagation time from the source to the recipient, which in itself is a difficult task because it requires that the time in all devices in the network should be synchronized. There are different time synchronization algorithms [7, 8], but all of them for synchronization use the time-codes (or packets), which propagate through the same links and have the same time delivery jitter as the interrupt-codes have.

If the time in a system is not synchronized then for measuring of interrupt-code propagation time it is required to allocate the additional testing interrupt-code types with acknowledges. Sending the testing interrupt-code and get acknowledge it is possible to estimate the interrupt-code propagation time to the recipient. For collecting the statistics it is necessary to send such tested interrupt-codes many times that will create a significant additional load of the network and also takes one or more interrupt-codes types.

These ways for solution of the interrupt-code delivery time synchronization task will be difficult and expensive in resources; It does not give the required accuracy also. Therefore it does not make sense to solve the interrupt-code delivery synchronization task by statistical methods.

SYSTEMATIC APPROACH TO THE INTERRUPT-CODE DELIVERY SYNCHRONIZATION

As was mentioned earlier and shown in equation (4), the interrupt-code delivery jitter is very dependent on network structure and links speed. The jitter will be greater if the recipient nodes are placed from the source at different distance, separated by a different number of routers, and if the speeds in links differ significantly. So if for some particular tasks in the network accuracy of interrupt-code delivery synchronization are important, then this task it should be solved at the system design stage.

The system structure should be chosen in a way to reduce difference in distances between the source node and other recipient nodes of interrupt-codes. All onboard networks for reliability and fault tolerance have redundancy. This redundancy should be added in a way, that when some device (routers or links) fail, t the length of the interrupt-code propagation path due to its broadcast would not change, or change insignificantly. It allows minimizing changing of interrupt-code delivery jitter. In the paper we considered the problem of interrupt-code synchronization delivery and described the ways for its solving:

- At the stage of system design we recommend to choose the network structure in a way, that when some routers or links fail,the length of the interrupt-code broadcasting propagation paths due to its are not changed, or changed insignificantly. Further, if it necessary, the correction values could be calculated theoretically for every recipient.
- If the the network changes during its operations insignificantly, than it is sufficient to use only static method of interrupt-code delivery synchronization with the pre-calculated correction values.
- If during the network operation its structure is changed significantly and the Plug and Play mechanisms are used, than we recommend to use data about all changes in the network from the manager of Plug and Play and, basing on these data, recalculate the correction values as for a static system and transmit new values to the recipients by RMAP command.

To use the methods with statistic collection in onboard networks is not rational. For a small network with the interrupt-code time delivery jitter will not be great because of paths length and speeds will not be differ significantly. If the network size is big then the process of statistic collection will be very complicated and recourse-intensive in terms of memory, traffic and processing consumption that is unacceptable for onboard applications; furthermore statistical methods may not to give better accuracy in comparence to static methods.

REFERENCES

[1] Space Engineering; SpaceWire Links, nodes, routers and networks, ECSS-E-ST-50-12C, July 2008.

[2] Sergey Gorbachev, Liudmila Koblyakova, Yuriy Sheynin, Alexander Stepanov, Elena Suvorova, Martin Suess, “Distributed Interrupt Signalling for SpaceWire Networks”. Proceedings of the 5th International SpaceWire Conference. Gothenburg 2013. ISBN: 978-0-9557196-4-6.

[3] Liudmila Koblyakova, Yuriy Sheynin, Elena Suvorova, “Asynchronous hard real time signals transmission in embedded networks”, International journal of embedded and

real-time communication systems. 2014, v.5(4), October-December, pp.24-44.

[4] Khramenkova K, Fortyshev E., "Tasks of Decentralized SpaceWire-Plug-and-Play Algorithm", 17th conference of the Open Innovations Association FRUCT, Yaroslavl, Russia, 2015, pp. 287-290, Publishing house of research university of Innovation Technologies, Mechanics and Optics "ITMO", Saint-Petersburg, ISBN 978-5-7577-0493-7, ISSN 2305-7254

[5] Fortysherv E., Khramenkova K. "SpaceWire Network Support Algorithm as a Part of Decentralized Plug-and-Play Algorithm", 17th conference of the Open Innovations Association FRUCT, Yaroslavl, Russia, 2015, pp. 283-286, Publishing house of research university of Innovation Technologies, Mechanics and Optics "ITMO", Saint-Petersburg, ISBN 978-5-7577-0493-7, ISSN 2305-7254

[6] European Cooperation for Space Standardization, "Space engineering – SpaceWire – Remote memory access protocol," ECSS-E-ST-50-52C, 5 February 2010

[7] A. Sakthivel, J.Ekergarn, D. Hellström, S.Habinc, M. Suess; SPACEWIRE TIME DISTRIBUTION PROTOCOL IMPLEMENTATION AND RESULTS, p.19-24, 6th International SpaceWire Conference, Athens 2014

[8] H. Kopetz, "Real-Time Systems. Design Principles for Distributed Embedded Applications. Second Edition", Springer, 2011.

Basic Study of Non-Contact Connector for High-Speed Space Cable Transmission

Space Fibre, Short Paper

Hiroshi Itakura, Yoshihiro Akeboshi, Hirotoshi Yamada, Hisashi Yoshiko
Mitsubishi Electric Co.
Kamakura, Japan
Itakura.Hiroshi@ah.MitsubishiElectric.co.jp,
Akeboshi.Yoshihiro@ak.MitsubishiElectric.co.jp
Yamada.Hirotoshi@ab.MitsubishiElectric.co.jp
Yoshiko.Hisashi@dy.MitsubishiElectric.co.jp

Satoshi Ichikawa
Japan Aerospace Exploration Agency
Tsukuba, Japan
ichikawa.satoshi@jaxa.jp

Atsutake Kosuge, Masashi Haraguchi, Tadahiro Kuroda
Dept. Electronics and Electrical Engineering
Keio University
Yokohama, Japan
kosuge@kuroda.elec.keio.ac.jp
haraguchi@kuroda.elec.keio.ac.jp
kuroda@elec.keio.ac.jp

***Abstract*—The data transmission rate required for the earth observation satellites is in the order of Gbps because of the enhanced performance of the observation sensors. However, the data transmission rate is approaching to the capable limits due to the issues of increased loss caused by the connectors or wires connected. In addition, there is another issue of the transmission irruption due to the vibration during the satellite launch. The transmission technology using the non-contact connector which has high vibration tolerance against the lossy wire connection systems is proposed and this paper describes the advantages of the non-contact connector, consisting of Transmission Line Couplers (TLC). With the transmission characteristics analysis of the connector and the design concepts, analysis of actual measurement results reveal that 15 m cables transmission of 2.5Gbps and BER less than 10^{-12} is feasible.**

***Index Terms*—Transmission line coupler, Non-contact connector, WizardLink, Cable, Comparator.**

I. Introduction

Data processing components that are mounted in satellites must be small and light, having high data transfer rates, and high storage capacity [1]. The next generation of earth observation satellites will require data transmission rates to a maximum of 20 Gb/s and at least one terabyte of storage capacity [2]. However, the data transmission rate is approaching to the capable limits due to the issues of increased loss caused by the connectors or wires connected. These connections consequently have the mismatch of impedance and complication of systems ensued by the increase in parts and wires of wired connections. In addition, there is another issue of the transmission irruption due to the vibration during the satellite launch.

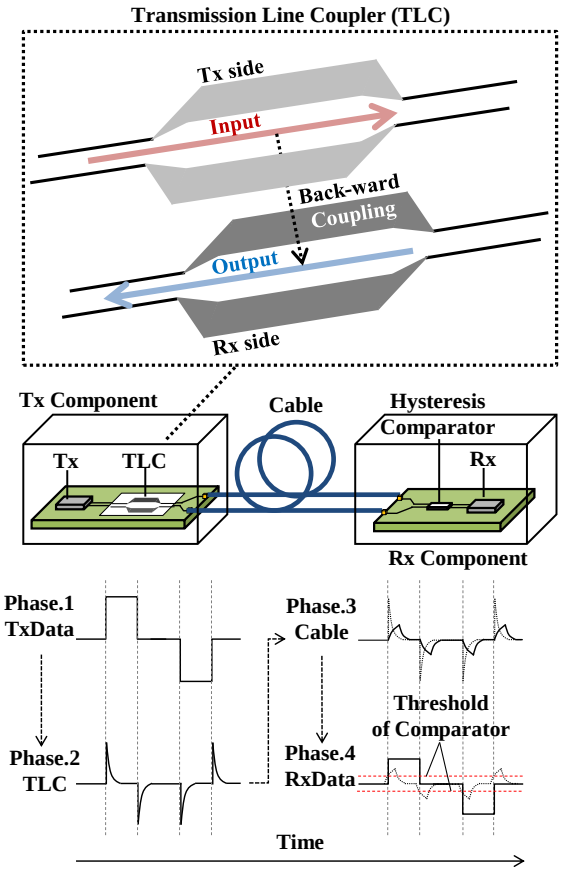


Fig. 1. High-speed space cable transmission system

In order to conquer the above problems, we focused our efforts on the new high-speed data transmission methodology, which is using the non-contact connector, consisting of Transmission Line Couplers (TLC) [3][4][5]. This method has

been known to enable high speed transmission up to 12.5Gbps and have high tolerance for vibrations during the satellite launch to the space. Moreover, it is considered that the transmission with the non-contact connector includes the potential to reduce the inter symbol interference (ISI) [4]. Thus our study suggested the transmission technology with non-contact connector for lossy wire connection system and this paper describes the advantages of that. With the 3D EM simulations and measurements with test boards, it could be possible that the proposed system could expand the length of cable transmission system up to 15 m of 2.5 Gbps and BER less than 10^{-12} is feasible with the tolerance for the vibration.

II. Cable Transmission System with Non-Contact Connector

The space cable transmission system with the non-contact connector is described in this chapter, shown in Fig. 1. Using the TLC as the non-contact connector in this system is the most important. Signals sent from the transceiver get into each TLC <phase.1>. After going through the TLC, the signals are shifted differential pulses because the TLC should behave the series capacitance which cuts dc components of the signals <phase.2>, and get into the cable, as lossy transmission line. Due to the conductivity loss of the long cable transmission, the pulses might be attenuated terribly when pulses reach the component where the receiver is mounted on <phase.3>; nevertheless the pulses could be reshaped to rectangle signals by the hysteresis comparator <phase.4>. A hysteresis receiver recovers the original data by retrieving them [5]. The recovered data signals are transferred to the receiver. Therefore, the receiver could get the high integrity data signals depended on the characteristics of the comparator however attenuated pulses might be. In addition, this system must be unrelated to the jitter caused by the ISI if the width of the pulse stays in a unit interval (UI), in principle.

III. Design Concept of Non-Contact Connector

A. Transmission Line Coupler as Non-Contact Connector

The design concept of the TLC as the non-contact connector is described by the view of EM/transient simulations in this section.

When a differential data signal is transmitted from a transceiver to the other module, its shape has to be changed to 1st order differentiated pulse shape in this system [5]. Therefore, it is necessary that the non-contact connector has low-cut characteristics. In order to satisfy the above characteristics, the differential TLC might be suitable and proposed in recent studies [3][4][5]. The TLC is made by use of simple board patterns without any connector components. The coupling range meets a dual constraint: not too long for ISI, and not too short for received amplitudes. The ac coupling is needed to be bilaterally symmetric, so that the signals and I/O circuits in both directions are the same [4]. The coupling distance is supposed that it could be adjusted by the thickness of the spacer whose dielectric constant is close to the substrate's dielectric constant. The TLC has been known to have the strong horizontal offset tolerance, which is about the same as the width of the coupler [5].

B. 3D Electromagnetic(EM) Analysis

3D EM model of the proposed TLCs in this study are shown in Fig. 2. The mixed s-parameters of this model are evaluated. The differential pulse is generally created by the backward coupling between the upper and the lower TLCs. Therefore, the mixed s-parameter from differential port 1 to differential port 2 (SDD21) is the critical parameter for the transmission characteristic of the pulse created by the TLCs.

The coupling gain of the TLC is known to be determined by the ratio of the electrode width (*W*) and the connection distance (*H*). The bandwidth and the amplitude are also known to be determined by the electrode length (*L*). The value of *W* is fixed to 3 mm, and *H* is fixed to 1.34 mm, as the typical design. The value of *L* is shifted from 6 mm to 10 mm by 2 mm steps in this section.

The result of 3D EM analysis is shown in Fig. 3. From the results, each TLC has the low-cut characteristics and broad bandwidth less than 10 GHz. Moreover, it is confirmed that the more *L* is longer, the lower the frequency at the peak level of the SDD21. These characteristics suggest that the TLC of which *L*=10 mm which has the largest gain at 1.0 ~ 5.0 GHz could generate a pulse whose shape has the largest amplitude and the pulse width.

C. Transient Analysis

Transient analysis model of the proposed TLCs are shown in Fig. 4. Transceiver's output parameters are set as TLK2711A, in order to compare the waveform of the pulse generated by the TLCs. The TLC model is the mixed s-parameter obtained in the preceding chapter. The data transmission rate is set 2.5Gbps as WizardLink transmission. As shown in Fig.5 (a) ~ (c), it is confirmed the conventional rectangle signals are certainly shifted to the 1st order differential pulses by the TLCs. In particular, it is clear that the pulse amplitude and width are the largest when the value of *L* is 10 mm. In this study, the TLCs of *L*=10 mm is accordingly selected from the perspective of the influence of the attenuation in the cable transmission system, and used for measurement works.

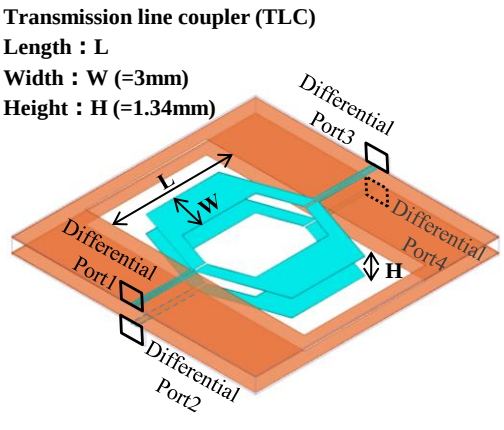


Fig. 2. 3D EM model of TLC

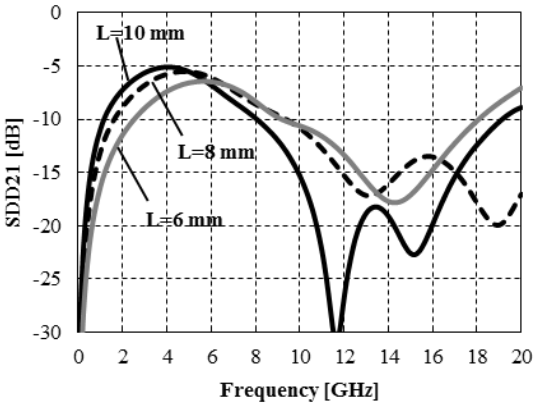


Fig. 3. Transmission characteristics of TLCs

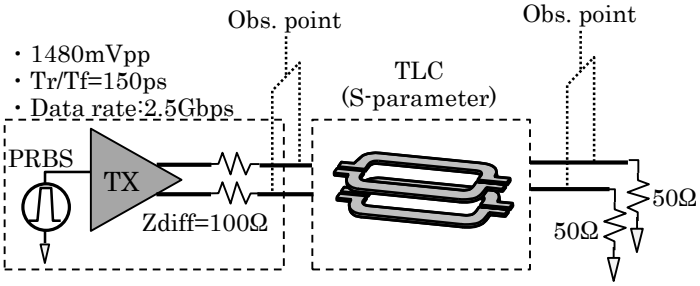


Fig. 4. Transient analysis model of TLC

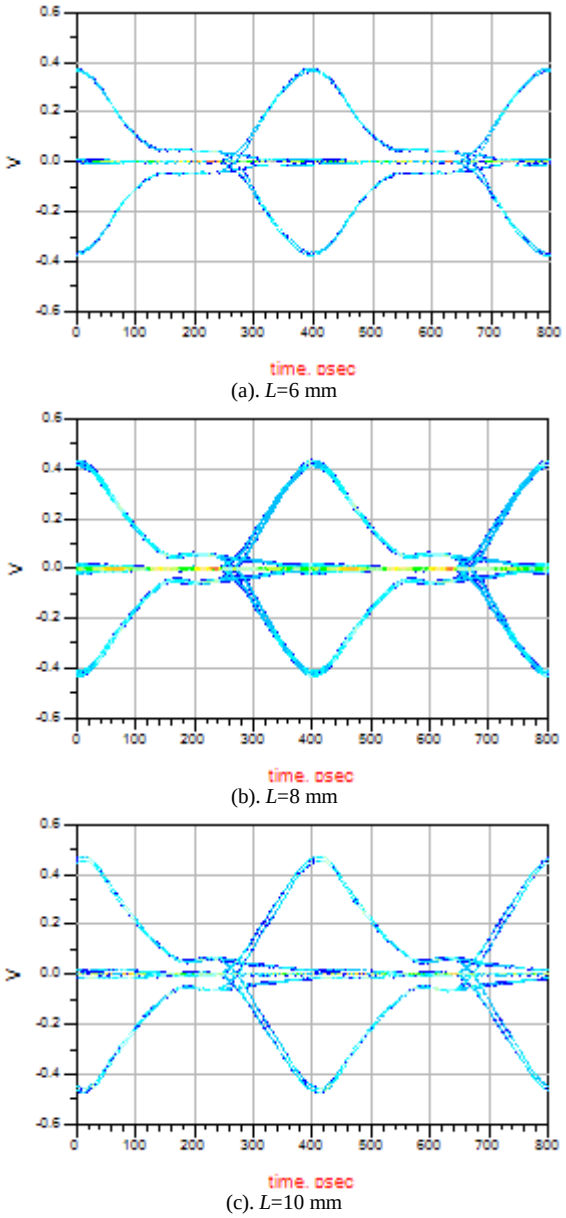
IV. Cable Transmission Measuremet

In order to confirm the availabilities of the proposed transmission system with the TLC, 15 m space cable transmission measurement is evaluated in this section. From the results of measurements, the validations of the design of the TLC and the signal integrity of the proposed system are discussed.

A. Measurement System

There are two measurement systems in this study, shown in Fig. 6. Time domain measurements were made with an Agilent DSO81304A oscilloscope at every probe point. Figure 7 shows the view of the measurement.

One is the proposed non-contact connector system, shown in Fig 6 (a). The TLC whose *L* is 10 mm was selected for this measurement because of its amplitude superiority for the lossy cable transmission. 1480 mV_{p-p} signals sent from transceiver, T LK2711A, get into non-contact connector, and are shifted to the differential pulses. The data transmission rate of this system is set 2.5 Gbps as the WizardLink transmission. Tx/Rx components are connected by the space cable whose length is 15 m. This cable called MW311 is made by Junkosha Co. The attenuation of this cable is defined 0.91 dB/m@1GHz. Then, differential pulses are reshaped to rectangle pulses by the hysteresis comparator. The levels of hysteresis thresholds are set ±70 mV, which could be estimated from the gain of the non-contact connector and the quantity of the decrement of the space cable.



(a). *L*=6 mm

(b). *L*=8 mm

(c). *L*=10 mm

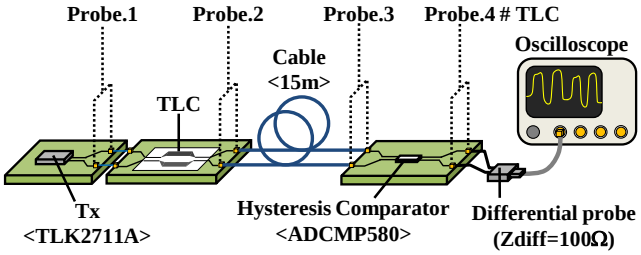
Fig. 5. Calculated waveform of pulse created by TLC

The other is the conventional system without the TLC in order to compare the transmission characteristics, shown in Fig.6 (b). Except the TLC and the comparator, it is the same setup with the non-contact connector system.

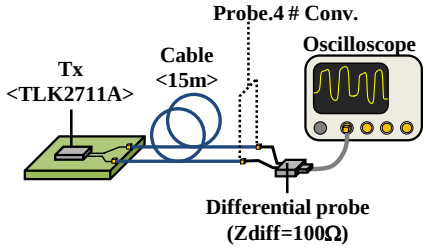
B. Results of Measurement

Fig.8 shows the results of the measurement at each probe point.

Probe.1 shows the eye pattern of the output data signals from TLK2711A. It is indicated that the output signals have approximately 1480 mV_{p-p} and 80 ps jitter. Thus, it could be said that the tendency of the waveforms of the transceiver model used in the transient analysis should be valid with that in this measurement system.



(a). Non-contact connector system



(b). Conventional system for comparison (without non-contact connector)

Fig. 6. System of evaluation for cable transmission

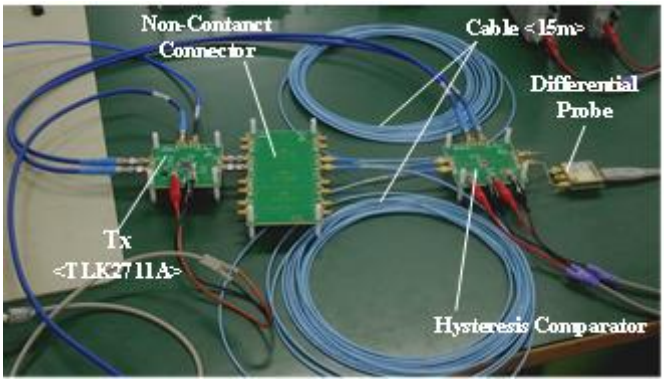
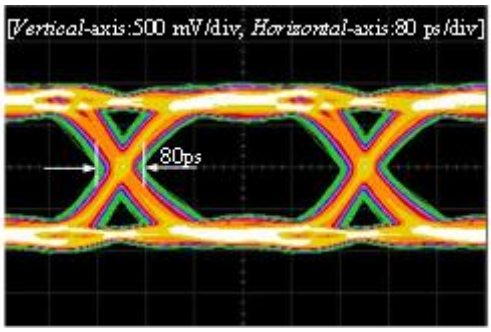


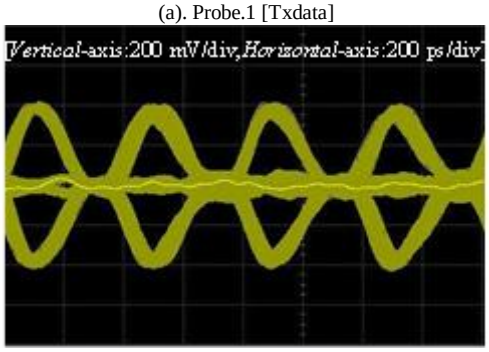
Fig. 7. View of measurement system

Probe.2 is the waveforms of the differential pulses created by non-contact connector. Shown in Probe.2, the differential pulses of about ±400 mV amplitude and less than 250 ps pulse width were obtained by the proposed TLCs. It suggests that the width of obtained pulses could be within less than UI [ps] at 2.5 Gbps.

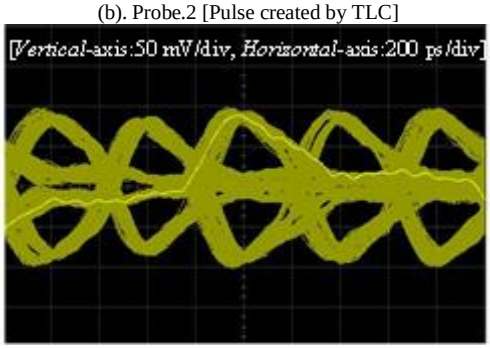
Probe.3 shows the waveforms of the differential pulses after transmitted through the 15 m cable. Due to the loss of the cable, the amplitude of the differential pulses were certainly attenuated by less than ±100 mV. Even though they might be attenuated by the lossy component, it is note that the comparator can reshape them to the rectangle pulse if the thresholds of the comparator are set properly. However, it is obvious that the width of differential pulses was force to be expanded to more than 400 ps because of the increase of the *RC* component of the cable. This width of the differential pulse might be over the UI at 2.45Gbps when it reaches to the comparator. Thus the waveform of Probe.3 seems to be affected by ISI, but it should be too small to cause the transmission irruption.



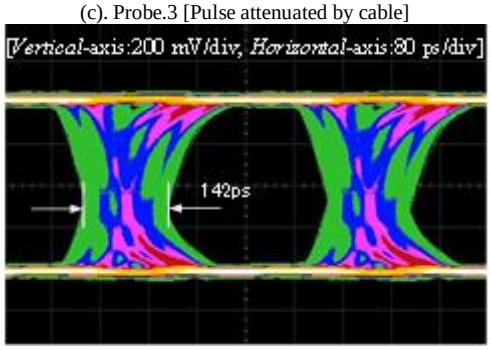
(a). Probe.1 [Txdata]



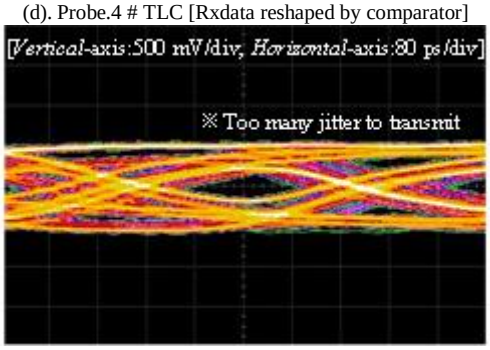
(b). Probe.2 [Pulse created by TLC]



(c). Probe.3 [Pulse attenuated by cable]



(d). Probe.4 # TLC [Rxdata reshaped by comparator]



(e). Probe.4 # Conv. [Rxdata without TLC]

Fig. 8. Measured waveforms

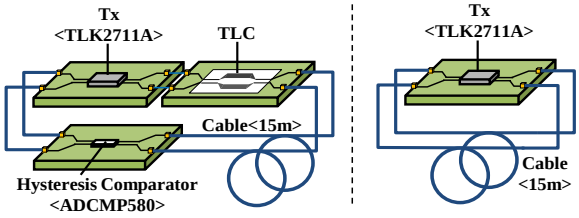
Finally, the eye pattern of the rectangle pulses reshaped by the comparator is shown in Probe.4#TLC. In order to confirm the availabilities of the non-contact connector system, the eye pattern of the conventional system is shown in probe.4#Conv. From these results, in the proposed non-contact connecter system, this clearly shows the differential pulses were reshaped well by the comparator and the signals transmitted at 2.5Gbps to the receiver. It suggests that the parameter of *L* of the TLCs and the hysteresis thresholds should be set properly taking the amount of attenuation of the cable into consideration. On the other hand, in Probe.4#Conv., it is confirmed that there is too many jitters to achieve the cable transmission in the conventional system. This is because the bulk conductivity and the loss tangent of 15 m cable simply influenced the eye opening.

C. Communication Test

Using embedded test function of TLK2711A, bit error checks were proceeded in order to confirm the integrity for the communications in the proposed system. Fig.9 shows the configurations of this test. The effective data transmission rates, is actually up to 2.0 Gbps, was transmitted. Consequently, It was confirmed that no signal bit failure occurred during the period of launch (1 hour). Therefore, the BER less than 10^{-12} for the WizardLink transmission was confirmed in the proposed system. On the other hand, it was observed that the communication test was end in failure continually in the conventional system. From the above, it could be said that the proposed non-contact connector system could have the availability for extending length of the transmission line, such as space cable, and so on.

V. Conclusion

The cable transmission system at 2.5Gbps with non-contact connector was presented in this paper. The detailed design methodology of the non-contact connector was described at first. The analysis suggests that the parameter of *L* of the TLCs determines the pulse amplitude and it should be set taking the amount of losses of the cable into consideration. In addition, analysis of actual measurement results reveals that the 15 m cables transmission of 2.5 Gbps and BER less than 10^{-12} is feasible with non-contact connector. It also means that the flexibility in the arrangement of the satellite components could be improved by using the proposed scheme. There is still room for improvement of broad band transmission characteristics for the shape of the differential pulse on the TLCs. Thus the alternative solutions to generate the more edged pulse with larger amplitude are needed, as the further study.



(a). Proposed system (b). Conventional system

Fig. 9. Configurations of Bit error check

Acknowledgment

This work was supported by CREST/JST.

References

[1] T. Seki, et al., "Development of Mass Data Storage for Space Applications with Commercial Memory Devices," 29th International Symposium on Space Technology and Science, 2013-j-21, June 2013.

[2] A. Kosuge, et al.., "A 6.5Gb/s Shared Bus Using Electromagnetic Connectors for Downsizing and Lightening Satellite Processor System by 60%," ISSCC, Dig. Tech. Papers, pp.434-435, Feb, 2015.

[3] A. Kosuge, et al., "A 12.5 Gb/s/link non-contact multi drop bus system with impedance-matched transmission line couplers and dicode partial-response channel transceivers," in Proc. IEEE Custom Integr. Circuits Conf., Sep. 2012, pp. 7.9.1–7.9.4.

[4] T. Simon, et al., "A 1.6Gb/s/pair Electromagnetically Coupled Multidrop Bus Using Modulated Signaling," ISSCC Dig. Tech. Papers, pp.184-185, Feb. 2003.

[5] A. Kosuge, et al., "Analysis and Design of an 8.5-Gb/s/Link Multi-Drop Bus Using Energy-Equipartitioned Transmission Line Couplers," IEEE Trans. Reg. Papers, vol. 62, no. 8, pp.2122-2123 Aug, 2015.

High Performance Network Components for Scalable Spaceborne Processing Needs

Poster, Short Paper

Joseph R. Marshall, Richard W. Berger

BAE Systems
Manassas, VA 20112
joe.marshall@baesystems.com

***Abstract*— this paper will describe high performance interface building blocks, compare their networking features and show how they may be used in small and large systems especially as they apply to SpaceVPX modules. Emphasis will be placed on their SpaceWire and other networking capabilities.[1]**

***Index Terms*—SpaceWire, Networking, Spacecraft Electronics, SpaceVPX, RapidIO, Endpoint, Router**

I. INTRODUCTION

Future spaceborne systems will require additional onboard processing and much greater interface connectivity. Many efforts worldwide are starting to address these needs. SpaceVPX, a recently released ANSI/VITA standard, was created to provide the structure and definition for interoperable modules that will be created to meet these needs. It provides a multi-layer set of fabrics using SERDES, LVDS and LVCMOS devices to provide interconnects in a scalable and fault tolerant way. Initial fabrics used by SpaceVPX are RapidIO, SpaceWire and I2C. Provisions are provided for heritage or user defined interfaces to interact with these within the structure. SpaceWire is setup as both a control plane for command and data handling throughout the box as well as a medium speed data plane.

Building on previous SpaceWire network elements such as its SpaceWire ASIC and its application specific standard products (ASSP) SpaceWire Endpoint ASSP (RADNET™ SpW-EP) and Golden Gate ASSP (RADNET™ SpW-RB4), BAE Systems is creating a set of silicon ASSP devices to provide power efficient general purpose building blocks for the creation of scalable SpaceVPX modules across these three fabrics. These building blocks are key to a new family of SpaceVPX processing and network modules being developed for a wide variety of space applications. 16 SpaceWire ports and a router are provided on the RAD5545™ multi-core system on a chip (SoC) and the RAD5515™ single core SoC while four SpaceWire ports and a router will be provided on the RapidIO Endpoint and the RADSPEED™ Host Bridge devices. These complement the higher performance four four-lane RapidIO ports on the SoC and Host Bridge devices, the one redundant four-lane RapidIO port on the RapidIO Endpoint (RADNET™ SRIO-EP), the 16 lane SERDES cross-point switch ASSP (RADNET™ 1616-XP) and the 48 RapidIO lanes on the RapidIO packet switch ASSP (RADNET™ 1848-PS).

[1] Approved for public release ES-ISR-080916-0097 ATR 467

II. INTERFACES

SpaceWire is a versatile interface fabric and now has over seven years of flight experience [1]. It was the first extendable standard fabric interface for spacecraft onboard processing and interconnection. It may be easily created out of space-worthy FPGA or ASIC components that support LVDS interfaces and thus has seen widespread usage as highlighted at six SpaceWire conferences. Its 200-400 Mbps bandwidth per link, enables high performance command and data handling as well as medium speed data movement. In future systems it is likely to continue to supplant PCI as a medium speed backplane interface. It is complemented by SERDES-based fabrics such as SpaceFibre and RapidIO, providing an order of magnitude bandwidth improvement per lane when needed. Taken together these support flexible and scalable heterogeneous systems and spacecraft.

III. HERITAGE SPACEWIRE ASSPS

In March of 2003, the first SpaceWire router ASIC [3] was started as a joint development project of BAE Systems and NASA Goddard Space Flight Center (GSFC). The four port SpaceWire router core as designed by GFSC was attached to the BAE Systems On-Chip Bus (OCB) through a router interface (RIF) block with two interfaces to the bus. The radiation-hardened by design (RHBD) ASIC in 250nm CMOS technology included on-die low-voltage differential signaling (LVDS) drivers and 64 deep FIFOs on each transmit and receive port. The ASIC included a BAE Systems 32-bit RISC microcontroller [4] called the embedded microcontroller (EMC) that can be used to program or interpret SpaceWire operations, two 16KB on-die SRAM blocks, a memory controller for external memory, and dual peripheral component interface (PCI) buses for connections to the rest of the module or system. The EMC is supported by a C compiler, and a software development environment that includes an assembler,

linker, debugger, and simulator. The "SpaceWire ASIC" was first flown on the NASA Lunar Reconnaissance Orbiter (LRO) mission, launched in 2009 [1] [2].

This device was followed by the decision to further integrate components in 150nm radiation-hardened CMOS technology, combining the standard bridge ASIC for the RAD750® radiation-hardened PowerPC™ processor [5] with both a four port SpaceWire router and a MIL-STD-1553 port as shown in Figure 2. Called the "Golden Gate" bridge, this RHBD application specific standard product (ASSP) introduced new features to the SpaceWire core, including four internal RIF interfaces allowing bypass of the router by all four ports if desired and the remote memory access protocol (RMAP) providing direct network access to the entire 4 GB address space of the device. An enhanced version of the EMC processor was incorporated with additional instructions, four 32 KB SRAM blocks, allowing for more on-die code and greater scratchpad memory storage. An enhanced memory controller was also included. This enhanced product is now being delivered with the newest generation of RAD750 processor flight modules. The ASSP is also available separately as part of BAE Systems RADNET™ family of products, where it is designated the RADNET SpW-RB4.

With the use of SpaceWire routers established and interest in the community to extend the SpaceWire standard to more parts of the system, a smaller SpaceWire endpoint ASSP was designed to allow existing instruments and peripheral functions to move to the emerging SpaceWire standard [6]. With a single redundant RMAP-enabled SpaceWire link, the SpaceWire endpoint offers a variety of alternative parallel and serial connections to existing designs, including I2C and SPI. The EMC processor is included along with a 32 KB SRAM and external memory controller. The RADNET SpW-EP is built in 150nm radiation-hardened technology using a RHBD circuit library.

IV. RAD55xx™ SoC ASSP Variations

The next generation of technology, a leap to RHBD 45nm silicon-on-insulator (SOI) CMOS known as RH45™ technology, offered the opportunity to develop a massively integrated high performance processor system-on-chip (SoC) as shown in the die layout in Figure 1. Based on Power Architecture® and leveraging the NXP (formerly Freescale) QorIQ® multicore communications processor family, an entire series of products was defined built on a platform RAD55xx™ ASSP [7] that can be configured into multiple personalities. The RAD55xx platform uses licensed intellectual property (IP) from the NXP P5020 and P5040 processors and other IP.

Supporting up to four 32/64-bit RAD5500™ processor cores, three levels of on-die cache memory, hardware encryption accelerator, dual interleaved DDR3 DRAM memory controllers, an SRAM/EEPROM controller, a Flash memory controller, up to four four-lane (x4) RapidIO ports @ 5 Gbaud/lane, and more, the RAD55xx platform also incorporated a 16-port SpaceWire router with RMAP and up to eight internal ports as shown in the block diagram in Figure 3. The sixteen port SpaceWire router is supported by the currently announced RAD5545™, RAD5515™, and RAD5510™ processors. A fourth product variant known as the RADSPEED™ HB processor, a host/bridge matched with the BAE Systems RADSPEED DSP [8], offers a 4-port SpaceWire router. Low speed interfaces including both I2C and SPI are also provided. The RAD55xx products have completed first hardware and are currently in test and characterization.

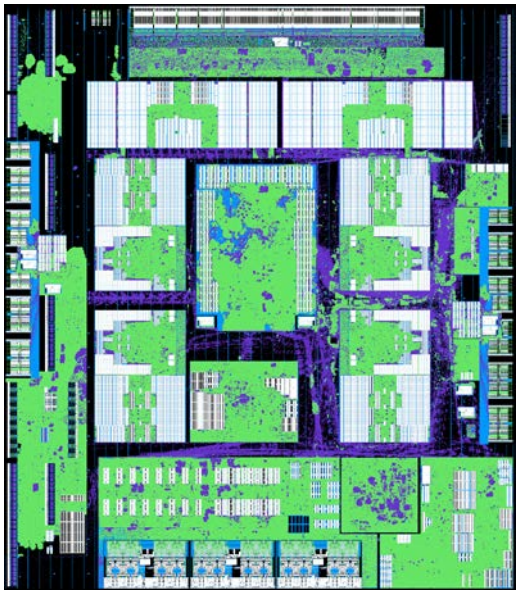


Figure 1: Die layout of the RAD55xx platform ASSP

The large SpaceWire router supports the ability to use these processors as the hub of a large SpaceWire data network or as the system controller of a high performance on-board processing subsystem based on the new VITA 78 SpaceVPX standard [9] [10] [11]. The SpaceVPX standard employs the RapidIO ports as the data plane and the SpaceWire links for the control plane. The SpaceVPX utility plane is supported with four I2C ports.

V. RapidIO Packet Switch

With the development of a high performance processor underway with both RapidIO and SpaceWire capability, the next logical step was to develop a RapidIO packet switch ASSP as the hub of the data plane for the on-board processing system. Based on licensed IP from Integrated Device Technology (IDT), the RADNET 1848-PS [12] is a RapidIO switch with up to 18 total ports of various widths or up to twelve x4 ports, all operating at up to 3.125 Gbaud/lane. The RADNET 1848PS, manufactured in 45nm SOI CMOS technology with the RH45 library, is now in hardware and is completing testing for delivery. Since this was almost completely based on purchased commercial IP, there is no SpaceWire port on the packet switch. Configuration takes place over an I2C interface which could be driven by a SpaceWire device such as the SpaceWire Endpoint ASSP or derived from the I2C interface on the SpaceVPX backplane.

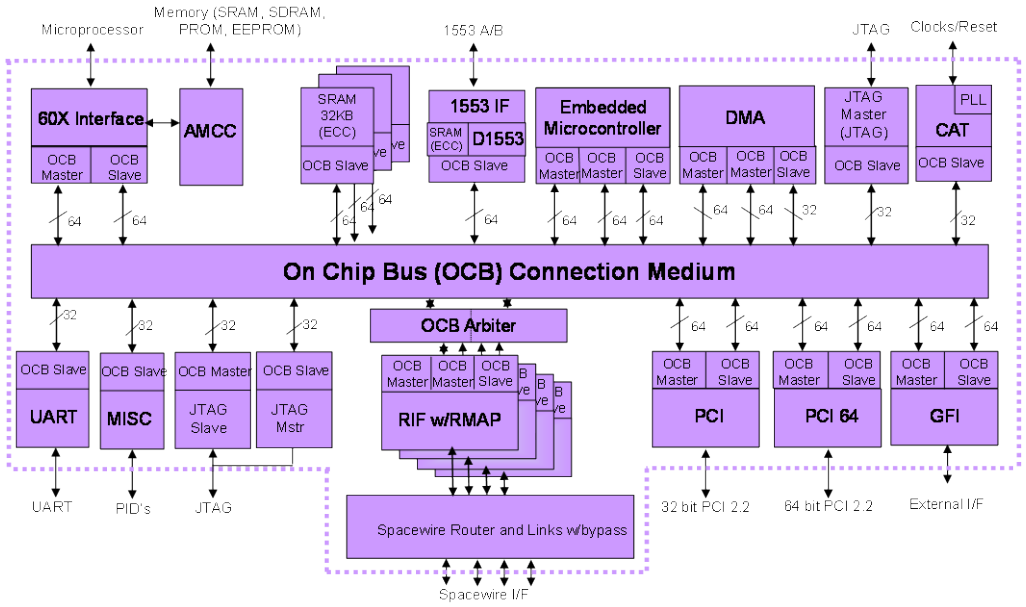


Figure 2: RADNET SpW-RB4 block diagram

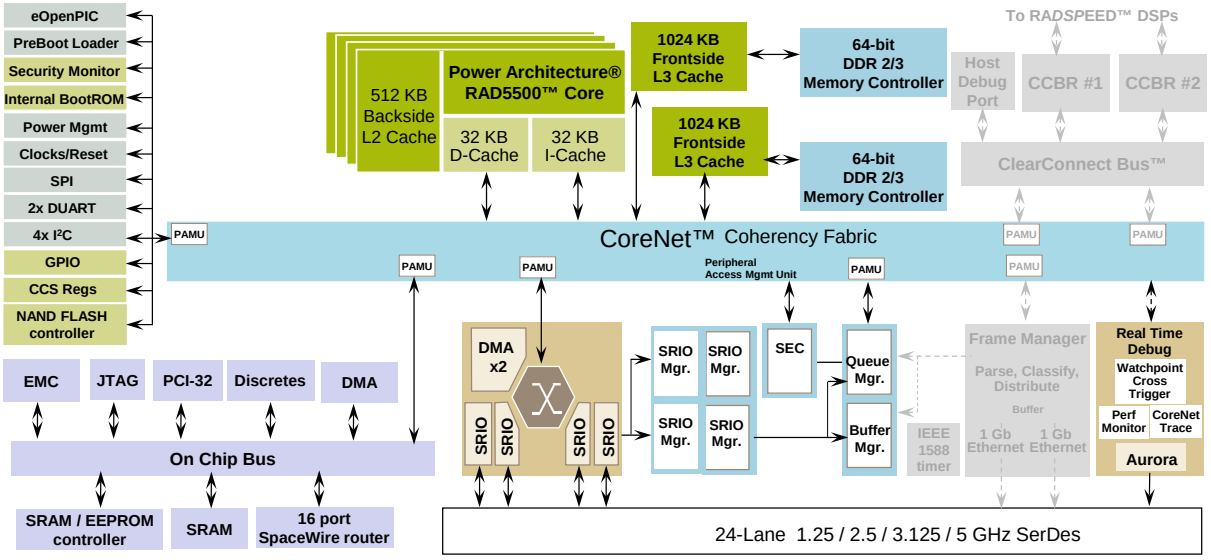


Figure 3: RAD55xx platform configured as the RAD5545 processor personality

VI. RapidIO Endpoint

Currently in design is a RapidIO Endpoint ASSP designated the RADNET SRIO-EP, also built in RH45 45nm SOI CMOS technology. It provides a single redundant RapidIO port @ 5 Gbaud/lane and a variety of additional ports including two redundant XAUI ports, a 32-bit PCI parallel bus, a redundant MIL-STD-1553 interface, a dual interleaved DDR3 DRAM controller, an EEPROM memory controller, a Flash memory controller, and both I2C and SPI serial ports. The RADNET SRIO-EP includes a four-port SpaceWire router with the same features found in the RADNET SpW-RB4 along with two 64KB blocks of SRAM. A simplified block diagram of the endpoint ASSP is shown in Figure 4. The RADNET SRIO-EP is designed as primary interface for SpaceVPX payload modules, with flexibility to provide the interface to mass memory (DDR3 or flash), or most other payload functions and interfaces through FPGAs or other ASICs.

From a processing perspective, the RADNET SRIO-EP includes both the EMC processor and four instantiations of a specialized version of the core called the SEMC. The SEMC core includes a 32KB instruction SRAMs that can be loaded with a specific program and a 24KB data scratchpad memory. The SEMC attaches directly to the ARM ABMA extensible

interface (AXI) bus. The four SEMC cores are supported by a 128KB block of on-die SRAM. As in the case of the EMC, the SEMC is supported by a software development environment and compiler. In the RADNET SRIO-EP, the four SEMCs work with a 4KB storage and hardware assist core called the "scoreboard" that accelerates priority queue management functions that are more typically performed in software [10]. The scoreboard hardware tracks transmitted and received word counts, identifying which queues contain data and when a queue reaches a programmed threshold.

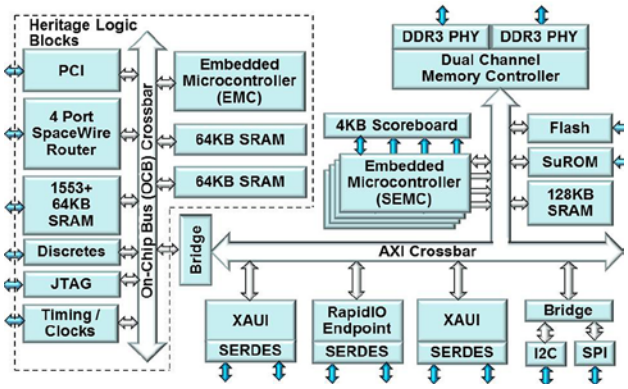


Figure 4: Block diagram of the RADNET SRIO-EP RapidIO endpoint ASSP

VII. Cross-Point Switch

Another component for use with high speed serial link networks is the RADNET 1616-XP ASSP. The protocol agnostic product provides for circuit switching of serializer-deserializer (SERDES) lanes at up to 5 Gbaud/lane with low latency and low power dissipation. It provides the ability to execute primary/redundant switching of SERDES signals, and also performs a repeater function through recovery of a degraded SERDES "eye" for longer distance transmission.

VIII. Systems Usage

Figure 5 shows a SpaceVPX system with several representative module types focused on using SpaceWire for Control and Data. This system is based on an application example in the revised SpaceWire standard and controls six instruments attached to the SpaceVPX chassis. SpaceWire is used as the control plane as well as a medium speed data plane. The controller uses its 16 port router to control and move data between all other logic modules. Shown in green are the BAE Systems ASSPs used to provide the SpaceWire interface functions. Not shown are additional FPGA or ASIC resources for unique functions or interfaces. A single string solution could be created using all the solid modules. Redundant modules are shown and dashed lines connect these to the other modules. The SpaceVPX star configuration supports 14 SpaceWire links on the backplane equivalent to up to 4.5 Gbps of cross sectional data bandwidth.

Figure 6 shows an upgraded system where RapidIO is used for the data plane and SpaceWire continues to function as the control plane. Here many of the SpaceWire components have migrated to RapidIO components that also support SpaceWire interconnects. The Mass Memory now relies on RapidIO on the data plane for its data stream inputs and outputs. Note the data plane switch is implemented in a seventh logic module. This system still provides the 4.5 Gbps of control plane cross sectional bandwidth over SpaceWire. The 12 ports of the data plane add an additional 120 Gbps of cross sectional bandwidth.

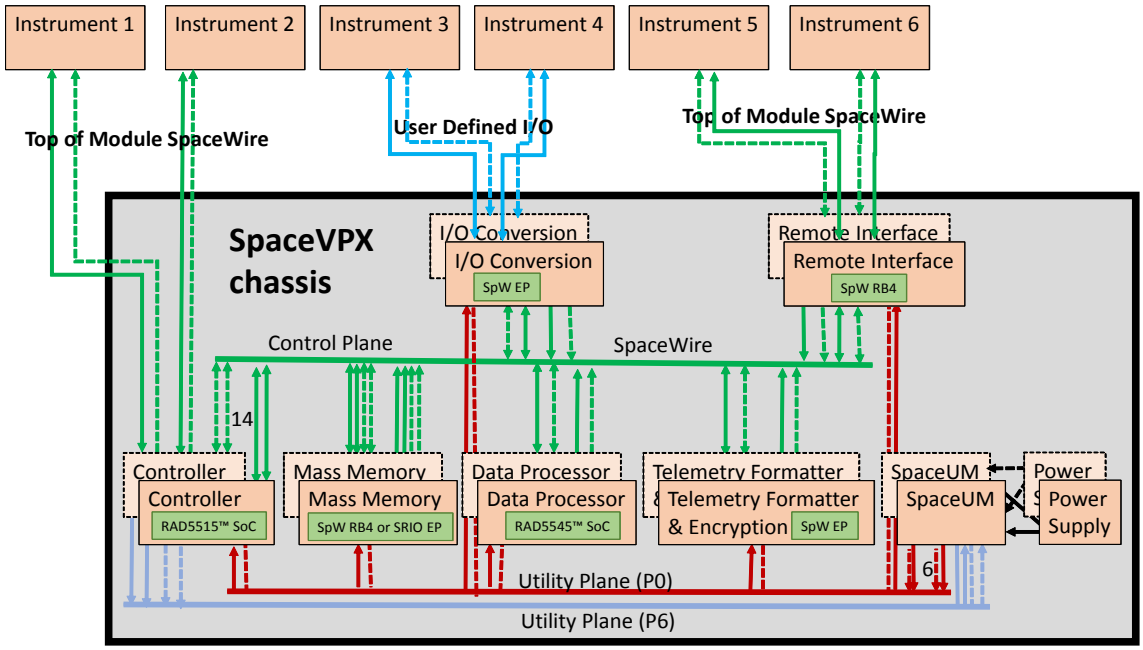


Figure 5: SpaceVPX system using SpaceWire for control and data. Solid horizontal lines represent point to point backplane connections.

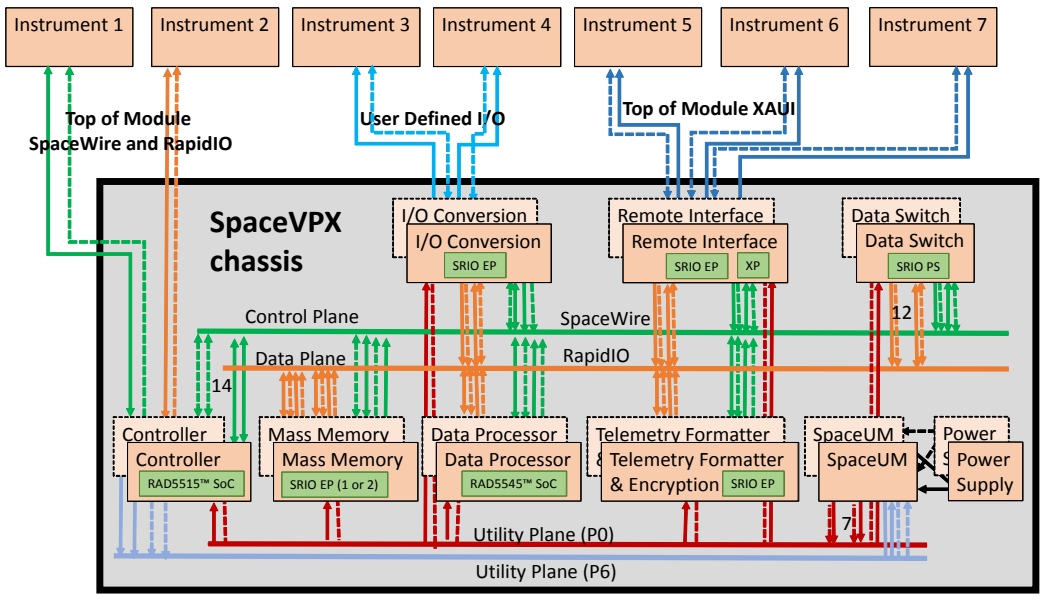


Figure 6: SpaceVPX system using SpaceWire for control and RapidIO for data

IX. Summary

BAE Systems has been actively developing and delivering SpaceWire solutions to the space community since 2003. These are transforming solutions from heritage bus-based modules to fabric-based solutions leveraging the recently ratified SpaceVPX standard for interoperable modules. Recent RAD55xx SoC and RADNET ASSPs under development and testing extend the fabrics to enable control, command and data handling solutions spanning four orders of magnitude of performance and the full range of typical digital functions in onboard spaceborne electronics. Using these devices and the SpaceVPX standard, families of scalable and interoperable modules may be created to meet current and future onboard processing needs.

References

[1] Berger, R. W., et. al., "RAD750 SpaceWire-Enabled Flight Computer for Lunar Reconnaissance Orbiter", Proceedings of 1st International SpaceWire Conference, Dundee, Scotland, September, 2007.

[2] Marshall, J. R., "Evolution and Application of System on a Chip SpaceWire Components for Spaceborne Missions", 2nd International SpaceWire Conference, Nara, Japan, 2008.

[3] J. Marshall and R. Berger, "A One Chip Hardened Solution for High Speed SpaceWire System Implementations", International SpaceWire Conference 2007, October 2007

[4] J. Marshall and J. Robertson, "An Embedded Microcontroller for Spacecraft Applications", IEEE Aerospace Conference 2006, Big Sky MT, USA, March 2006

[5] R. Berger, et al, "The RAD750 – A Radiation Hardened PowerPC Processor for High Performance Spaceborne Applications", IEEE Aerospace Conference 2001, Big Sky MT, USA, March 2001

[6] J. Marshall, S. Santee, M. Hanley, J. Robertson, D. Stanley, "Leveraging SpaceWire Networking Prototyping to Create Flexible SpaceWire Components and Support Software", International SpaceWire Conference 2011, San Antonio TX, USA, November 2011.

[7] R. Berger, et al, "Quad-Core Radiation-Hardened System-on-Chip Power Architecture Processor, IEEE Aerospace Conference 2015, Big Sky MT, March 2015

[8] J. Marshall, R. Berger, A. Berard and M. Bear, "Applying Advanced Networks and Signal Processing to Spaceborne Computing", AIAA Infotech@Aerospace 2012, Garden Grove CA, USA, June 2012.

[9] Collier, Charles Patrick, et al., "Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space", Proceedings of 2015 IEEE Aerospace Conference, Big Sky, MT, March 2015

[10] J. Marshall, "Standardized SpaceWire Solutions for Next Generation Systems", Proceedings of the 2014 International SpaceWire Conference, Athens, Greece, September 2014.

[11] P. Collier, J. Marshall, R. Berger, M. Enoch, S. Goedeke, "Next Generation Space Interconnect Standard (NGSIS): A Modular Open Standards Approach for High Performance Interconnects for Space", AIAA 8 Reinventing Space 2013 Conference Proceedings, Los Angeles, CA, September 2013.

[12] D. Rickard, et al, "On-Board Networks with Radiation-Hardened 45nm SOI Standard Components", IEEE Aerospace Conference 2015, Big Sky MT, March 2015

All figures in this paper are Copyright BAE Systems and used with permission.

SpaceWire Test Center in Japan

SpaceWire test and verification, Short Paper

Iwao Fujishiro, Shigeyuki Arase
Shimafuji Electric
8-1-15 Nishikamata,Ota-ku,Tokyo144-0051,Japan
fujishiro@shimafuji.co.jp

Masaharu Nomachi
Osaka University
1-1 Machikaneyama,Toyonaka,Osaka560-0043,Japan

Shoichiro Mihara, Kenji Sasaki
Japan Space systems
3-5-8 Shibakoenn Minatoku, Tokyo 105-0011 Japan

Abstract— **In 2006, we developed SpaceWire platform named SpaceCube cooperation with JAXA and NEC. After the success of SpaceCube project, we developed number of SpaceWire products. Some examples of this innovation include several kinds of the SpaceWire interface boards, SpaceWire router and SpaceWire-to-GigabitEtherR2. These developments included the support and cooperation of JAXA, OSAKA University, Japan Space Systems and NEC. However, there are the big step into the Space market for the small high tech companies. In this paper we describe Renewed SpaceWire Test Center in Japan.**

***Index Terms*—SpaceWire, Test**

I. Introduction

The SpaceWire Test Center is open to the public who are interested in SpaceWire study and development. The engineers have to prepare own testing environment to study and develop own components, equipment etc. However, this is the barriers for the small organizations who are considering or develop the SpaceWire components.

Shimafuji Electric Inc. has opened The SpaceWire Test Centre in Tokyo to the public who need to test their own components to adapt SpaceWire. This paper describes the background, purpose, use of images and configuration of the SpaceWire Test Center.

II. The Backgrounds

Shimafuji Electric Inc. joined ASNARO consortium and started the SpaceWire test facility in 2010. At that time, it was a small scale test facility, but since then has helped test products created by consortium members.

Shimafuji upgraded this test facility in 2012, however, this upgrade was designed for limited projects. Recently in 2016, this facility became public open testing center.

III. The purpose

Reduction of the barriers

It is possible to lower the barriers for engineers or teams. Shimafuji manufactured and installed an advanced high-speed / high-performance test equipment and capable facilities for flight model test.(Fig1, Fig2)

Fig.1. Clean booth

Fig.2. Constant temperature and humidity chamber

Promote the SpaceWire (Tutorial environment)

Shimafuji developed the manual for SpaceWire Test procedure. All new entry member, who are students, engineers of organization or manufacturing companies, can understand and prepare environment condition, hardware and software for the SpaceWire testing. We also developed the low-cost SpaceWire tutorial system for SpaceWire beginners and researchers training. (Fig3, Fig4)

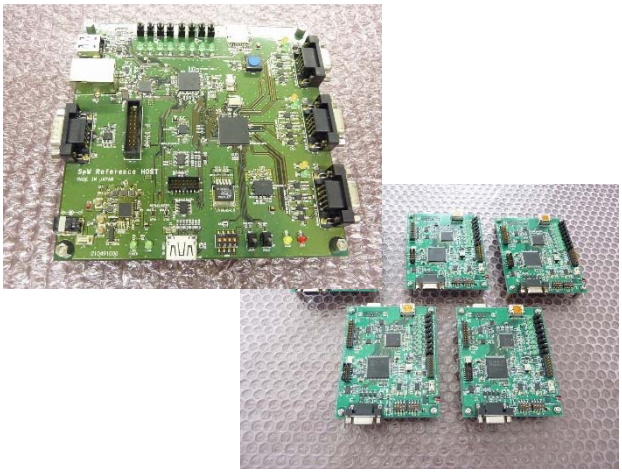


Fig.3. Tutorial system.

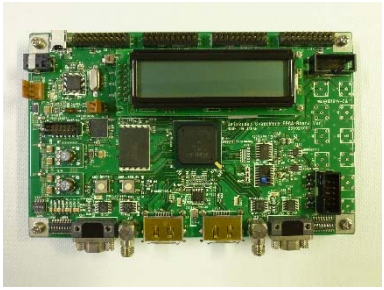


Fig.4. Universal SpaceWire FPGA Board.

Improvement the Open SpW IP quality

The Open SpaceWire FPGA IPs and the Test Scripts were released and maintained.

IV. The Use Case

Connecting Test

The Center has the equipment to evaluate SpaceWire connection with Conformance Tester, and SpW RMAP Tester. The Center also fitted with debugging tools, Link Analyzer, and Logic Analyzer FPGA tools. (Fig5, Fig6)

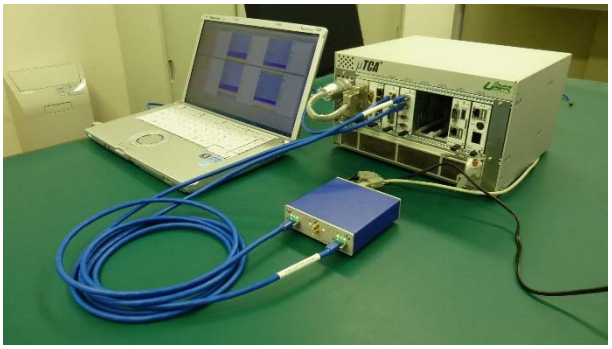


Fig.5. Analyzing the packets.

Fig.6. SpW backplane 12slots

Physical layer evaluation and Environment test

The center has the Network Analyzer and Digital Oscilloscope for Physical layer evaluation as well as temperature and humidity chambers for environment testing. In conjuncture with these testing environments, a clean booth for flight model. There are special equipment to evaluate the SpaceWire cables performance. (Fig7)

Fig.7. SpaceWire cable Test.

V. The Test Center Configuration

TABLE1 The test center configuration

Classification	Device and equipment
SpaceWire equipment	Conformance Tester(STAR-Dundee) Link Analyzer(STAR-Dundee) Router USB(STAR-Dundee) Multilink Analyzer(4Links) SpaceWire-to-GigabitEtherR2(Shimafuji) SpaceCube(Shimafuji) SpaceCubeMK2(Shimafuji) 6PortRouterUnit(Shimafuji) SpW DIO II (Shimafuji) SpW backplane 12slots(Shimafuji) SpW RMAP Tester(Shimafuji) Tutorial system(Shimafuji) Open IP(Shimafuji)
SpaceFiber	Universal SpaceWire FPGA Board (Shimafuji) High speed SpaceWire Flash ADC Board (Shimafuji)
Instrument Development tools	Network Analyzer(Agilent) Logic Analyzer(Agilent) Digital Oscillo(Agilent) SpaceWire Adapter(Agilent) FPGA tools (Xilinx/Altera/Actel)
Environmental testing	Clean booth (3m x 3m, class 1000) Constant temperature and humidity chamber

VI. Space Technology to Ground

Shimafuji developed the SpW-R board which use daisy chain connection for less wiring, and based on SpaceWire and RMAP. This idea is wide use of SpaceWire into industry and commercial. This board has 3 port router and I/Os which can control motor, camera and etc. The board is reasonable for non-space industries. (Fig8, Fig9) *This board and demo system will be in the SpaceWire test center.

Fig.8 SpW-R board

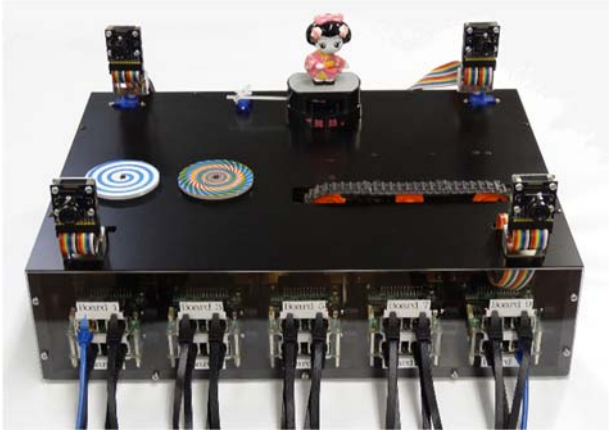


Fig.9. SpW-R Demo system

VII. Conclusion

The Center has wealthy instruments to Test SpaceWire in Tokyo and it also possible to provide consultation of the SpaceWire introduction and technical support.

This center will accumulate feedbacks from user to improve the open IPs, and will work as Japanese SpaceWire information source. The center is expected to increase number of SpaceWire users in coming years.

References

[1] Makoto Ioki, Takeshi Tohara(Japan Space Systems), Iwao Fujishiro(Shimafuji Electric Incorporated),Masaharu Nomachi(Osaka University), Seisuke Fukuda, Tadayuki Takahashi (JAXA/ISAS) ,"SpaceWire Test Office: Practical examples", Ukaren, Beppu, November 2012

[2] Shigeyuki Arase, Iwao Fujishiro(Shimafuji Electric Incorporated), "SpaceWire-to-GigabitEther and SpaceWire backplane", Shimafuji Electric, International SpaceWire Conference, Athens, September 2014.

A NEW MISSION DATA RECORDER (MDR) WITH TIME-SEARCH FUNCTION FOR ERG MISSION SYSTEM

NOT PERMITTED TO PUBLISH PAPER

SPACEWIRE NETWORKING SYSTEM FOR PAYLOADS ONBOARD ERG SATELLITE

NOT PERMITTED TO PUBLISH PAPER

DEVELOPMENT OF REAL-TIME AND HIGH-SPEED SPACEWIRE DATA TRANSFER SYSTEM

NOT PERMITTED TO PUBLISH PAPER

SPACEWIRE ELECTRICAL TESTING

NOT PERMITTED TO PUBLISH PAPER

Multi-purpose simulator for Plato mission

SpaceWire mission and applications, Short Paper

Rafael Corsi Ferrão, Sergio Ribeiro Augusto, Cássio Berni, Franklin Ronald Ferreira dos Santos e Vanderlei Cunha Parro
Critical embedded systems group
IMT
São Caetano do Sul, SP, Brazil
corsiferrao@gmail.com

Saulo Finco
Citar project
CTI
Campinas, SP, Brazil
saulo.finco@cti.gov.br

Philippe Plasson and Loïc Gueguen
LESIA
Paris Observatory
Meudon, France
Philippe.plasson@obspm.br

Gisbert Peter
DLR
Berlin, Germany
Gisbert.Peter@dlr.de

Manfred Steller
IWF
Graz, Austria
Manfred.Steller@oeaw.ac.at

***Abstract*— This paper presents hardware and software solution for simulation of a group of cameras used by the PLATO satellite. The simulator can be configured either to work with the scientific processing unit or as the complementary loop attitude control processing unit. Configuration and monitoring can be performed remotely, which caters to cooperatively work and guarantees traceability for quality purposes. Because the system operates with database and configurable architecture, its performance can be modified to operate as standard generation element (CCSDS, e.g.) traveling via SpaceWire. Eight SpaceWire links for feeding processing system compose the simulator. The links may route data in real time rate configurable to 200Mbits/s, with representative images, using the RMAP protocol, it can run continuously up to two days of satellite operation. The information database is stored in two solid-state drives with 500 GBytes capacity each one. Access for configuration and monitoring are made using TCP-IP protocol. Each simulator has a unique ID and is automatically recognized when connected to the Ethernet network. The software layer has graphical interface compatible but offers component for integration with other EGSE systems. The system has own housekeeping, which enables diagnosis operation and viewing by the operator. The system will be used by European groups: LESIA (France), DLR (Germany) and IWF (Austria).**

***Index Terms*— SpaceWire, RMAP, Plato mission.**

I. Introduction

The main goal of this project is to have a realistic hardware simulation of the image acquisition system of the Plato satellite. Part of this architecture can be analyzed in Fig. 1. Each data processing unit (DPU) receives from the electronic front end (FEE) 4 SpaceWire (SpW) [3] links running at 100 Mbits/s. A similar system is used in the attitude control system, just changing the number of links to 8 and reducing the data amount to a half CCD. The simulator described here involves the CCDs, with dynamic images and the FEE. More details about Plato architecture is available in the ESA website [1].

The simulator can be subdivided in three subsystems:

- Electronic main board (EMB).
- Resident software (RSW).
- Supervisor software (SSW).

The EMB has the capability to work with eight SpW links and to generate all RMAP [2] commands necessary to transmit the image from the simulator to the DPU and to receive RMAP commands (Write and Read) from the DPU.

The RSW software controls EMB where the images are stored, running on a dedicated PC and communicates with the EMB by a S-ATA protocols. The set RSW+EMB forms the Simulator (SIMUCAM).

Supervisor software (SSW) is proposed to control a set of simulators and communicates with the RSW by a TCP/IP connection allowing debugging and configure several simulators. The simulator runs as stand-alone application to avoid timing glitches and the Human-Machine-Interface (HMI) will be use only to control the simulations.

The SIMUCAM is able to work with both: normal DPU (N-DPU) used for scientific processing and fast DPU (F-DPU), used for the attitude control loop. They work in a similar way besides the integration image time and the number of SpW interfaces.

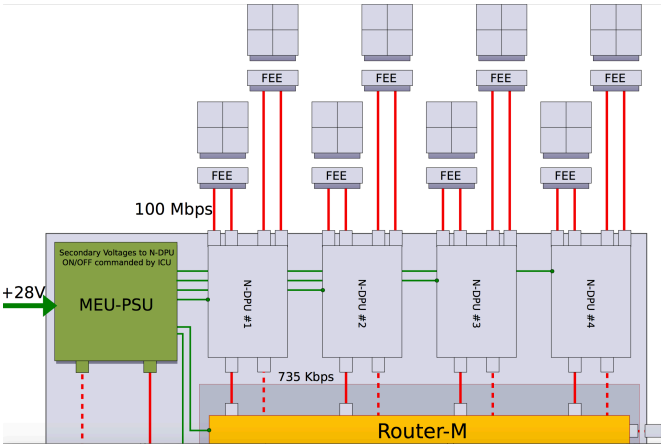


Fig. 1. Plato electrical architecture for science.

A. N-FEE description operating in normal mode

Each N-FEE is in charge of four CCDs. Each CCD is acquired sequentially, the period of activity for normal camera is 25 seconds, The full process (4 CCD) takes 25 seconds. During these 25s, a full image from one CCD is delivered to N-DPU each 6.25s. The full-image transfer time from N-FEE to N-DPU should take considering the maximum optimization value of the time transfer less than 3.3 seconds. Two SpaceWire link is implemented as communication protocol between N-DPU and N-FEE.

The CCDs are divided vertically in two parts (Left and Right), and each part is transmitted by one of the two links. The CCDs transfer are swapped each 6.25 seconds.

For safety reason the SpaceWire channel utilization shall not exceed 80 %, that means the data rate averaged over the transfer duration, including the SpaceWire overhead, shall not exceed 80 Mbps. N-FEE is connected to a N-DPU by two SpaceWire links, each link is responsible for transfer half of CCD each CCD at time. The transfers occurs by encapsulating half line of one CCD (Left and Right) on a RMAP write command (FEE to DPU) and send it by one of the links, as shown on Fig.2 The address of each write commands is incremental and should be restarted at a new image transfer (25 s).

Each write command (IMAGE) have at the begining of the data a top sync counter that is incremented at a new received synchronism, and at the end of the data some *prescan* pixels. The *prescan* pixels can be part of the image (loaded into the simulator) but the top sync counter must be added by the MEB on real time. The top sync (6.25s) is transmitted to de DPU by a *timecode* command; this is the only way that DPU can access this signal. It will be sent by both SpW links to guarantee robustness. Housekeeping is sent by the N-FEE to the N-DPU at the end of a full image transmission by a RMAP write command (N-FEE to N-DPU). The HK can also be accessed asynchronous at any operation mode by a read command from the DPU (N-DPU to N-FEE).

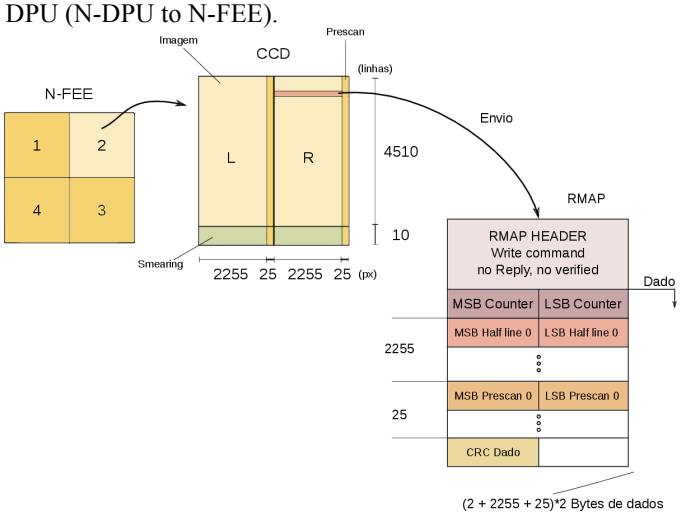


Fig. 2. RMAP data diagram.

The N-FEE has the following functional modes:

- Operational mode: the CCDs are read with the synchronization signals. Data packets including image and Housekeeping are sent to N-DPU.
- Stand-By mode: no sequencing signals and no data packet sent. Bias to CCD at nominal value, only Housekeeping data are sent on request from the MEU.
- Integration mode: during AIT specific read must be done to compensate the huge dark current generated by the CCD, the use of the Dump-Drain or a different exposure time management must be implemented. In that mode, the N-FEE may function without synchronization signals from the *SylBox*.
- Test Mode: the N-FEE sends a pattern to the MEU. This mode is mainly used during AIT step with the MEU without the FPA. This pattern will be defined latter.

B. F-FEE description operating in fast mode

As the N-FEE each F-FEE is in charge of four CCD. Each CCD is acquired sequentially, the period of activity for the fast cameras is 2.5 seconds, The full process (4 CCD) takes 2.5 seconds. During these 2.5, all full images are delivered to the F-DPU each 1.5s. Eight SpaceWire link is implemented to transfer the images to de DPU. The CCDs are divided vertically in two parts (Left and Right), and each part is transmitted by one of the eight links. The CCDs transfer concurrently (all at the same time). F-FEE is connected to a F-DPU by eight SpaceWire links, each link is responsible for transfer half of CCD, this process occurs concurrently. The transfers occurs by encapsulating half line of one CCD on a

RMAP write command (FEE to DPU) and send it by one link, as shown on Fig. 2 The address of each write commands is incremental and should be restarted at a new image transfer (2.5 s) this means that the images are saved on the same memory address at the DPU side.

C. ICU

There are 2 ICU channels, which work, in cold redundancy. The ICU is responsible for the management of the payload, the communication with the Service Module (SVM), the compression of scientific data before transmitting them as telemetry to the SVM. Two SpaceWire routers (RU) a Data Compression Unit (RDCU), a memory unity (MU) and a processor unit (PU) compose the ICU.

Each ICU Router Unit (RU-A and RU-B) is connected to

- 4 MEU (Router Unit A is connected to MEU routers A and Router Unit B to MEU routers B).
- 2 F-DPU
- 4 N-AEU
- 2 F-AEU

Other functional units of ICU throughout 4 additional SpW links:

- ICU internal Memory Unit (MU)
- Processing Unit A (PU-A)
- Processing Unit B (PU-B)
- The other Router Unit to connect together the two SpW network.

The RDCU will collect the data from the twelve front end DPUs and compress the data. Finally, the ICU generates the telemetry packets to be sent to ground.

II. SIMULATOR DESCRIPTION

The global vision of the Simulator is synthesized in the Fig. 3 where we can see the components that compose the project. In order to understand the MEB role in overall process it is important to establish the main interaction during a nominal operation. The MEB is responsible to transmit data (ack. Images) from its internal SSD and DDR memory to the SpaceWire links, encapsulating it on a RMAP protocol. All transfers are started by a synchronism pulse (sync) that can be external our internal. With the sync detected, the processor core starts the tasks responsible to load the images stored on the DDR2 memory to the FIFOs on each SpaceWire/RMAP peripheral. Eight different DMA controllers, each one associated to a specific link, perform this transfer.

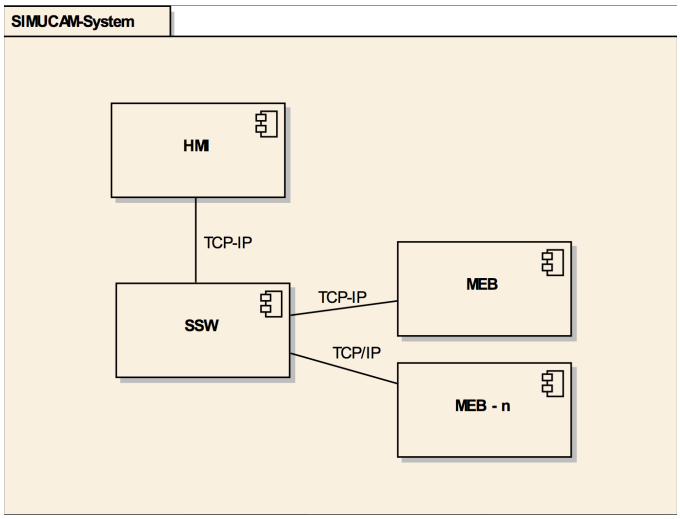


Fig. 3. SIMUCAM system and its components.

In order to understand the MEB role in overall process it is important to establish the main interaction during a nominal operation. The MEB is responsible to transmit data (ack. Images) from its internal SSD and DDR memory to the SpaceWire links, encapsulating it on a RMAP protocol. All transfers are started by a synchronism pulse (sync) that can be external our internal. With the sync detected, the processor core starts the tasks responsible to load the images stored on the DDR2 memory to the FIFOs on each SpaceWire/RMAP peripheral. Eight different DMA controllers, each one associated to a specific link, perform this transfer.

The DMA core supports a buffer of 512 transfers each transfers cares 9020 bytes. At the beginning of transfer the transfer command buffer is full filled by the task that controls the specific link. During the transmission of the image, the DMA core generates a interruption at every DMA transfer executed, this causes the link task to fill again the command buffer.

The 8 DMA cores share the DDR2 memory, the round robin scheduling is used to give access to DDR2, this scheduling is performed, not on software level, but at hardware level (bus controller).

The SpaceWire/RMAP FIFO is of 32 bits wide (to be compatible with the Avalon bus) and has a depth of 512. This peripheral is responsible to fragment the data into RMAP packets and transmit it to the SpaceWire link. No software innervation is necessary once the peripheral is configured, its perform the auto increment of the destination address on the RMAP command, inserts the data CRC, inserts (if configured) extra data on the data (ID info, top counter).

All this is done to free the uC from critical tasks the MEB avoiding glitch on the data transfer. Figure 4 is a resume from the proposed architecture. The SSD are used to stored images, this images must be loaded by the Ethernet TCP/IP connection

that communicates with the SSW component. These images are than cached on the DDR2 memories, and transfer by each SpaceWire link respecting all specification imposed for the N-FEE and F-FEE.

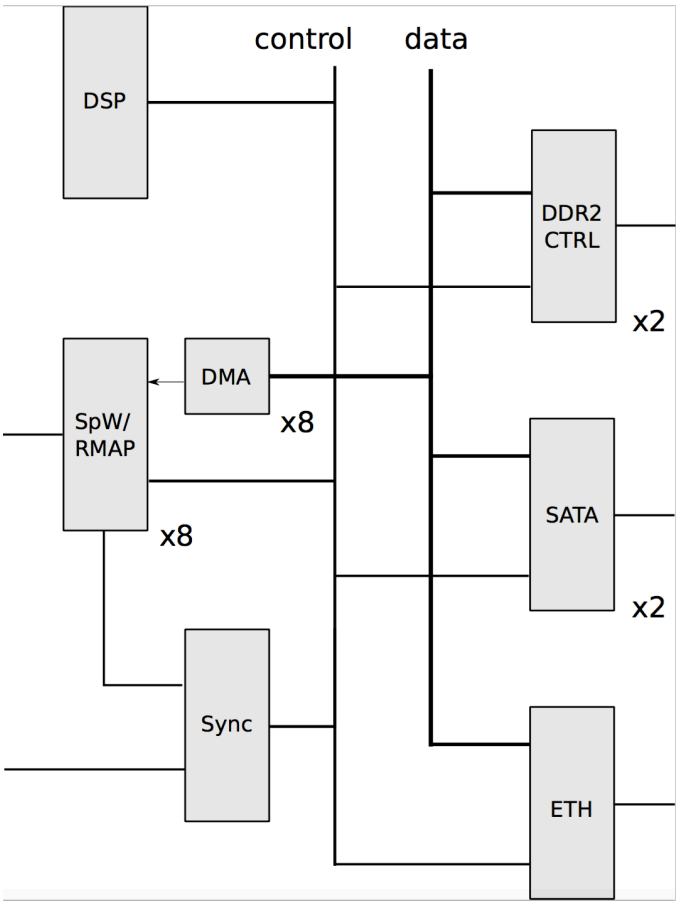


Fig. 4. The simulator architecture.

The SpW/RMAP is peripheral developed for optimize the data transfer from de memory to the SpaceWire, which can be analyzed in Fig. 5, its six main parts can be listed:

- RMAP: generates the RMAP head and data CRC; its operation is controlled by the REG.
- FIFO: used to cache de data of the RMAP (image), is a dual port with a reading clock of 200 MHz (8b) and writing port of 100 MHz (32b).
- DELAY: block that generates the sample time (4MHz) to emulate the A/D .
- REG: register with all RMAP configuration.
- TC: Time code.
- SpW: SpaceWire core.

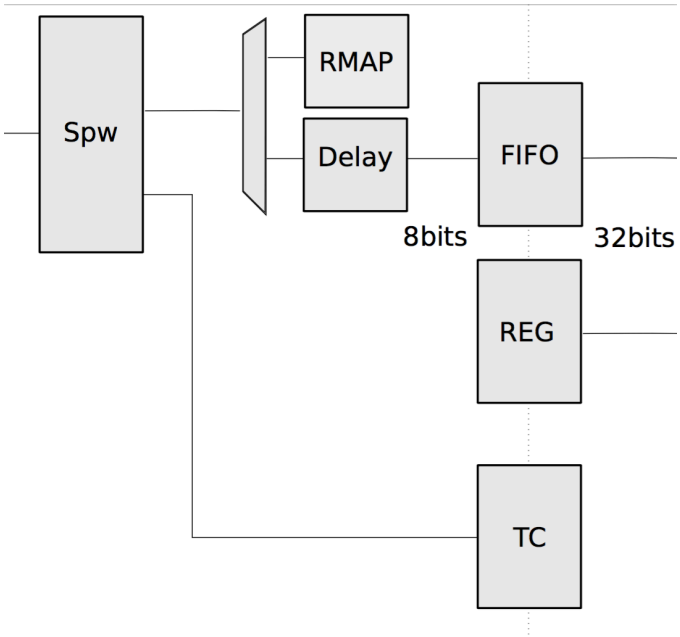


Fig. 5. The SpW/RMPA peripheral architecture.

ACKNOWLEDGMENT

The authors thanks to: Fapesp grant 2008/57866-1; CNPq grant 574004/2008-4; FINEP and Mauá Institute of Technology.

REFERENCES

[1] ESTEC-ESA. Plato payload definition document. ”http://sci.esa.int/ science e/www/object/index.cfm?fobjectid=42793#, 2008.

[2] ESTEC-ESA. Remote memory access protocol remote acess memory protocol. http://www.ecss.nl/forums/ecss/dispatch.cgi/standards/showFile/100770/d2010020912165 E-ST-50-52C(5February2010).pdf, 2010.

[3] ESTEC-ESA. Spacewire protocol identification. http://www.ecss.nl/forums/ecss/dispatch.cgi/standards/showFile/100769/d2010020912161 E-ST-50-51C(5February2010).pdf, 2010.

Radiation tolerant heterogeneous Multicore "system on chip" with built-in multichannel SpaceFibre switch for onboard data management and Mass Storage Device

Components, Short Paper

Tatiana Solokhina, Jaroslav Petrichkovich, Alexander Glushkov, Andrey Belyaev, Leonid Menshenin, Fedor Putrya
ELVEES RnD Center,
Zelenograd, Russia,
tanya@elvees.com

Sheynin Yuriy, Suvorova Elena
University of Aerospace Instrumentation,
St. Petersburg, Russia
sheynin@aanet.ru

***Abstract* — The article presents a 180nm CMOS Radiation tolerant heterogeneous Multi-core ASIC MCT-04 as the SoC (System–on-Chip) with built-in multichannel multiprotocol SpaceFibre/ GigaSpaceWire (SpaceWire-RUS standard), SpaceWire based switch for the onboard data and Mass Storage Device management. The SoC design and architecture support Single-Event-Upset (SEU) fault-tolerant. The MCT-04 embedded networking subsystem provides multiple ports for high-rate interconnection with combination of SpaceWire/ /GigaSpaceWire (SpaceWire-RUS)/SpaceFibre links. Input and processed data streams transmitted via 1.25 Gbps four multiprotocol SpaceFibre/GigaSpaceWire links with built-in DMA controllers. Two SpaceWire links (ECSS-E-50-12C) provide data transfer bandwidth 2 - 400 Mbps. The MCT-04 embedded networking subsystem on the base SpaceWire/GigaSpaceWire/SpaceFibre provides a balance between external and internal data throughput especially for the multifunctional micro and nanosatellites systems.**

***Index Terms* — Radiation tolerant heterogeneous Multicore ASIC, multiprotocol SpaceFibre based links, NAND-Flash, Memory Controller, on-board Mass Storage Device management**

I. INTRODUCTION

In the spacecraft data processing and storing systems, it is necessary to solve several important tasks, including:

1. Delivering large amounts of data at high speed from the sensors to the proper processing system;
2. High-speed data streams switching;
3. Storing of large amount of data;
4. The overall management of space system.

The article describes the experience in the creation of MCT-04 "system-on-chip" qualified for space application with architecture intended to provide high-speed data exchange between data source, data processing and data storage blocks, and also between multiprocessor networks on the SpaceWire, SpaceFibre and Giga SpaceWire (SpaceWire-RUS standard) base.

Thus, the limiting factor in the development of Mass Storage Device or SSD (Solid-state Drive) s the absence of a large selection of space microprocessors for the high-performance space computing, that provided the highly throughput by the links (up to the gigabits) based on modern advanced standards such as SpaceWire and SpaceFiber and its modifications.

This article describes the experience in the creation of an actual high-performance the highly throughput "system-on-chip" MCT-04 qualified for space applications with balanced architecture of processing IP-cores and network subsystem of the data exchange between ASIC resources and between multiprocessor networks on the SpaceWire, SpaceFiber [2] and Giga SpaceWire (SpaceWire-RUS standard) base.

II. THE MCT-04 ARHITECTURE

Radiation tolerant Multicore ASIC MCT-04 (Fig.1) was developed as the homogeneous SoC (System–on-Chip) for the onboard data and Mass Storage Device management.

The block diagram of the chip is shown in Fig. 2.

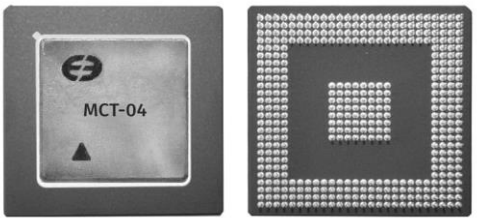


Fig. 1. MCT-04 chip in the CPGA720 package

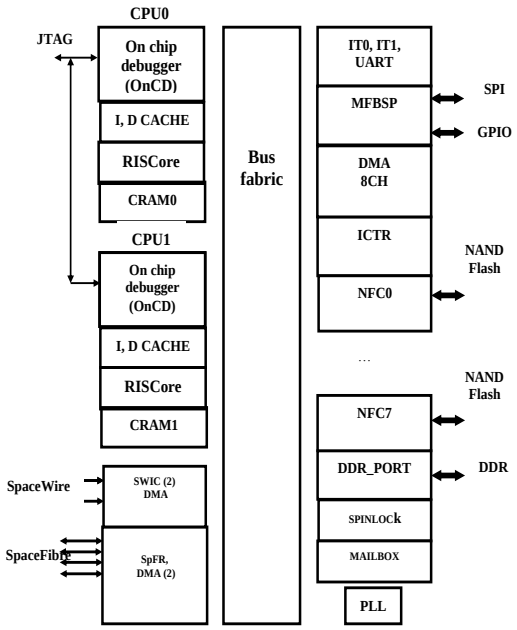


Fig. 2. MCT-04 chip block diagram

The main MCT– 04 SoC features are:

- CPU0, CPU1 – central processing units 0, 1 based on RISC cores with MMU, 4-stage pipeline and Multiply/Divide accelerator; CPU clock frequency is no less 120 MHz, which apply CPU performance as 120 MOPS each;
- On chip memory - about 3 Mb, including CRAM0, CRAM1 – random access memory of CPU 128 KB each;
- Error correction of internal and external memory : single error correction and detection of double errors Hamming code;
- I, D CACHE – instructions and data cache of CPU 32 KB each;
- DMA MEM_CH – 8-channels DMA;
- 32-bit DDR port, 1600MB/s for the external memory;
- Eight NAND Flash Controllers (NFC0:NFC7). Each of them supports speed ranging from 40 MB/s to 200 MB/s; provides a connection 128 NAND Flash chips totaling 2 Tbytes using 128 Gb components; also provides a connection 32 NAND Flash 3DPlus firms modules totaling 1 Tbytes
- SWIC0, SWIC1 – SpaceWire interface controllers;
- SpFR (SpaceFibre Router) - 4-port multiprotocol switch SpaceFibre/GigaSpaceWire (SpaceWire-RUS). The capacity of each port from 5 MBd to 1.25 GBd with RMAP protocols supporting. The connecting to AXI switch was realized via two multi-channel DMA controller;
- MFBSP - multi- buffered serial port operates in the controller mode SPI bus and GPIO[2:0];
- ICTR – interrupt controller;
- UART – universal asynchronous port;
- IT0, IT1 – universal timers, interval/real time;
- SPINLOCK – low-level mutual exclusion synchronization primitive;
- MAILBOX – messaging module;
- OnCD – built-in hardware debugging tools and JTAG – debug port.
- Power Saving Modes support;
- PLL – frequency multiplier PLL based;
- Power consumption - no more 3 W.

The ASIC consists of two processing CPU. CPU is a standard RISC - processor (RISCore32) with 4-stage pipeline. With Multiply/Divide accelerator CPU provides the addition, multiplication and division operations. CPU also has a memory management unit (MMU) on the basis of fully associative address translation buffer (TLB) of 16 double cells, the instruction cache (I CACHE) of 32 Kbytes of data cache (D CACHE) of 32 Kbytes. The programmable MMU provides two operating modes: with TLB (Translation Lookaside Buffer) and FM (Fixed Mapped). On-chip JTAG IEEE 1149.1 Debug Unit support the single stepping and data address/value breakpoints.

MCT-04 ASIC was realized to support all architectural solutions, which increased its resistance to failure and fault tolerance. All ASIC memory blocks including the register files in CPU/DSP are protected by Hamming code with single errors correcting and two errors detecting.

The MCT-04 applies the ability to turn off unused processor IP cores and other resources such as unused high throughput links. The MCT-04 also supports a sleep mode in which it consumes minimum milliwatts of power.

The ASIC has DDR memory ports (1600 MB/s), support DMA transfers between external I/O ports and external memory, have Multifunctional Buffed Serial Ports (MFBSP) that can act as SPI or GPIO interfaces, six Space Wire family.

Input and processed data streams via through six SpaceWire based family links (four up to the 1.25 Gbps and two up to 400 Mbps) provide a balance between its throughput and SoC performance.

MCT-04 also has a dedicated test and debug interface; run the Linux operating system; and have a C /C++ application software compiler for the CPU.

The NAND Flash Controller has an AHB Interface, which allows the CPU processor to configure the operational registers sitting inside the NAND flash Controller. The IP core supports the Open Nand Flash Interface Working Group (ONFI) 1.0, 2.0, 2.1 and 2.2 standard. The NAND flash Controller handles all the command, address, data sequences, manages all the hardware protocols, and allows the users to access NAND flash memory simply by reading or writing into the operational registers.

Features of NAND Flash Controller:

- Supports Flash devices up to 128 Gb
- Supports NAND Flash memories from Micron, Samsung, ST-Micro and others.
- Supports all mandatory commands and selected optional commands

- Boot mode support, Full access to spare area
- Supports speed ranging from 40 MB/s to 200 MB/s to allow applications to balance performance and Power
- Supports Interleaving Operations: Page Program Interleaving, Copy back Program Interleaving, Block Erase Interleaving, Read Interleaving, Cache Interleaving
- upports Multi LUN/Die Operations
- Supports Small Data Move
- Supports Change Row Address
- Supports Reset LUN: Page Size - 512B, 2KB, 4KB, 8KB
- Flash data bus width: Standard support - 8bit for both Asynchronous and Synchronous mode, Additional support - 16bit only for Asynchronous mode
- ECC:
  - Hamming Code: 1Bit error correction, 2Bit error detection
  - BCH: tandard support: 4, 8 bit error correction; Additional support - up to 32 bit error correction
- provides a connection 16 the Flash chip. Eight NAND Flash Controllers provides a connection 128 the Flash chip - a not less than 1 terabyte.

III. MCT-04 DESIGN ISSSUES

In addition to radiation tolerant and low power requirements for the space applications chip there was an additional requirement to achieve 2 GB/s exchange rate to NAND mass storage from external devices. During the preliminary analysis of the architecture, the following critical points that may affect the final performance of the system as a whole have been identified:

- SpaceFibre ports performance (digital part + PHY)
- NAND controllers performance
- DDR memory port performance
- On-chip interconnect performance
- CPU performance, in particular:
  - CPU load by processing of requests coming from SpaceFibre by RMAP protocol
  - CPU load with NAND management (processing the request queue, reorder memory accesses operations, interrupt from the NAND controllers handling)
  - CPU load by queries allocation algorithm between eight NAND controllers and software cache management to increase the overall performance and lifetime of the memory chips

Each of the given points can become bottleneck with critical impact on system performance. To avoid unnecessary system redesign cycles Requirement Driven Verification approach was applied [5]. Initial specification performance requirements continuously checked at all stages of design, from the very first stage in which there is uncertainty of the system architecture. In the route was used:

- Analysis of requirements to the system elements from memory management algorithm with high abstraction level TLM models;
- System efficiency analysis task decomposition (autonomous research of IP-Cores performance);
- Entire system performance analysis with RTL model simulation and complex system-level tests [6] [7].

The principal feature of the route was the feedback from every direction on all remaining studies. Thus, when more detailed models of IP-cores was created, it causes updates for TLM model environment, in which the memory management algorithm was debugged (e.g. the number of interrupts from the NAND controller). In other way, new details obtained from algorithm environment causes updates in interfaces bandwidth requirements and CPU performance. In particular, it was fundamentally important to obtain information on the service traffic from the CPU for memory management algorithm (in addition to main data traffic).

Here are some important results of research carried out in the process of designing the SoC

Case 1: 8 NAND controllers together create a huge stream of interrupts (8 interrupt per transmission of one 4K page), which is completely distracts the resources of a single processor with architecture selected for the project.

Two solutions were adopted for this reason:

- Place additional CPU in a system for controlling the eight NAND controllers
- Implement inside NAND controller hardware more intellectual management, for example for sending pages in auto-increment address mode (block transfer mode)

Together, these solutions have led to the release of CPU0 for cache and SpaceFibre management tasks and selecting CPU1 fully for NAND control task, which in a typical mode of operation has reserve for software optimization and parallelization of NAND transaction flow.

Case 2: System level complex tests analysis of on-chip interconnect and memory subsystem performance showed CPU traffic (arbitrary traffic with short transactions) crowding out effect by a continuous high-density traffic from NAND and SpaceFibre ports. In addition, DDR bandwidth achieved in such tests is very close to maximum, determined by given technology. Therefore, even the change of the arbitration scheme and buffer organization in the DDR controller would result in a loss of performance for NAND traffic, which would also reduce performance, but in another place. The simplest solution was to increase the internal memory size and place in it program and the most frequently used algorithm data. CPU traffic minimization has beneficial impact on system performance for mass storage use case.

TABLE I. THE CROWDING OUT EFFECT FOR CPU TRAFFIC BY NAND AND SPACEFIBRE HIGHER DENSITY TRAFFIC

master	Average time of byte transmission, ps
nand3_w	3318.68489583
nand2_w	3385.41666667
nand4_w	3447.265625
.....	
cpu1_r	29436.7913148
cpu1_w	32159.8223481

NAND controller and system performance:

TABLE II. NAND CONTROLLER PERFORMANCE

	ONFI mode performance	SLC mode read performance	SLC mode write performance
16-bit mode	400 MB/s	304 MB/s	160 MB/s
8-bit mode	200 MB/s	152 MB/s	80 MB/s

TABLE III. SYSTEM PERFORMANCE

	Required average interconnect bandwidth	Payload data traffic speed
4 KB packets traffic	375 MB/s	5-7 MB/s
256 KB packets traffic	350 MB/s	Up to 180 MB/s
mixed 4/256 KB packets traffic	308 MB/s	50-60 MB/s

SoC design analysis showed that depending on the nature of the external traffic system would show the performance from 5MB/s (traffic from the packet of minimum size at which the limiting factor is the CPU performance) up to 200MB/s.

200MB/s - practically achievable maximum for SpaceFibre in the SoC, but large amount of different packets may drop performance to 140-180 MB/s, caused by limitation in the DDR bandwidth in case of simultaneous streams from all masters.

Two CPU design left margin in terms of software optimization potential, thus lower performance bound can be improved in future work with firmware.

Large on-chip memory for memory management algorithm program and data in addition to cache eliminates negative impact of CPU activity on overall system performance.

The MCT-04 software platform apply the Complete tool set for the fast development and integration of the space applications, includes MCStudio ® IDE (Integrated Development Environment).

IV.THE MCT-04 ASIC EMBEDDED SPACE WIRE STANDARDS FAMILY NETWORKING SUBSYSTEM

The MCT-04 embedded networking subsystem provides multiple ports for high-rate interconnection with combination of the SpaceWire/SpaceFibre /GigaSpaceWire (SpaceWire-RUS standard) links.

The combination of the SpaceWire based family links (SpaceWire, SpaceFibre and GigaSpaceWire with various speeds and opportunities) provides unprecedented flexibility and scalability for space on-board processing systems.

The six MCT-04 SpaceWire based family serial high-rate links consist of:

1) Four multiprotocol ports (belong to SpFR switch) such as SpaceFibre (2 VC, 1250Mbps) /GigaSpaceWire (SpaceWire-RUS); have rates up to 5,10,15 ... (with 5 Mbps increments) ... 125, 312.5, 625, 1250 Mbps);
2) Two SpaceWire ports (ECSS-E-50-12C) have rates up to 2-400 Mbps.

It should be noted that MCT-04 ASIC SpaceWire links implementation supports the extensions towards next SpaceWire standard revision such as Distributed interrupts and others.

It is also important to note that the GigaSpaceWire ports can provide bandwidth up to the 1250 Mbps, but can operate also in a range of lower data rates, down to 5 Mbit/s. Lower data rates could efficient for longer distances or using older types of cabling.

GigaSpaceWire is in fact a high-rate link for SpaceWire networks, and has the exactly the same Packet, Network layers and the same packet formats that makes the packets routing and switching between any combination SpaceWire and GigaSpaceWire ports straightforward and resource-efficient. The internal switch operates as a SpaceWire routing switch, with routing and switching SpaceWire/GigaSpaceWire packets between any combinations of its ports, in accordance with ports operation modes and the routing table.

Two SpaceFibre links [3] are supporting by the multiprotocol network interface controller. The main SpaceFibre link rate in the MCT-04 ASIC is 1250 Mbit/s.

In the multiprotocol ports implementation another operation mode is to support the GigaSpaceWire protocol. Such combination of the different types of ASIC links (SpaceWire/SpaceFibre/GigaSpaceWire) and internal switches makes the MCT-04 very flexible in building ASIC network interconnection with external processors, nodes, and peripherals with any type of SpaceWire/SpaceFibre /GigaSpaceWire networks; provide different types of network services.

While SpaceFibre links provide advanced QoS features (very important for the space onboard systems), the SpaceWire/GigaSpaceWire combination links provide effective and cost-efficient networking for other on-board applications (for example, for the space Mass Storage systems). Such applications may not require SpaceFibre QoS features with an extra cost of the SpaceFibre implementation silicon area.

SpFR switch supports the following main types of information flow:

- Streams RMAP packets with hardware-software package processing;
- Flows package SPW, a software package processing; no hardware packet processing is not performed. The various transport protocols and application layer can support software for them;
- Streams RMAP appeals teams in the space SpFR block configuration from remote network administrator.
- Interpretation of flow types calls made in relation to the logical channel numbers SpFR virtual ports from which they come.

All packets coming from virtual channel VC0, automatically interpreted as RMAP treatment team in the configuration space.

For virtual channel VC1 is possible to program the settings of interpretation modes - either as a stream of packet transport protocols , either as a stream of packets SpW not interpreted SpFR unit and processed by software.

Thus MCT-04 ASIC is a new generation “system on a chip” of that supports a wide class of space on-board applications ranging from onboard data management to Mass Storage systems.

V. THE MCT-04 ASIC PROOF ON THE SILICON

In this MCT-04 ASIC project it was created a new innovative multiprotocol port IP-core (SpaceFibre/Giga-SpaceWire IP–core) that provide a balanced solution between all advantages in QoS, FDIR and others from SpaceFibre and the simplicity and low cost of implementation from GigaSpaceWire. In this MCT-04 ASIC project it was created a new innovative multiprotocol port IP-core (Space-Fibre/GigaSpaceWire IP-core) that provide a balanced solution between all advantages in QoS, FDIR and others from SpaceFibre and the simplicity and low cost of implementation from GigaSpaceWire.

MCT-04 ASIC was developed and synthesized on the space qualifiable ASIC technologies base. The chip size is 17.5 mm x 17.5 mm (Fig.3). During the project, we analyzed the complexity and feasibility of 4-channel SpaceFiber switch built-in microprocessor with two virtual channels each.

From the MCT-04 ASIC post-layout area analysis the size of the silicon area for some radiation tolerant MCT-04 IP - cores (real layout):

- CPU0, CPU1: 47.00 mm*2;
- 4-port multiprotocol switch SpaceFibre/Giga-SpaceWire (SpaceWire-RUS): 42.5 mm*2
- SpaceWire interface controller (SWIC): 2.5 mm*2;
- NAND Flash controller (NFC): 10.5 mm*2.

The main parameters of the SpaceFibre/GigaSpaceWire CML based transceivers IP-cores, based on the space qualification Radiation Tolerant Libraries, are:

- A wide range of data rates 5, 10, 15… (with discrete of 5)...125, 312.5, 625, 1250 Mbps – for the GigaSpaceWire mode (including multiprotocol links) and - 1250Mbps for the SpaceFibre mode;
- The transmitter and receiver IP blocks dimensions are the same: RX = TX = 0.233 mm*2.

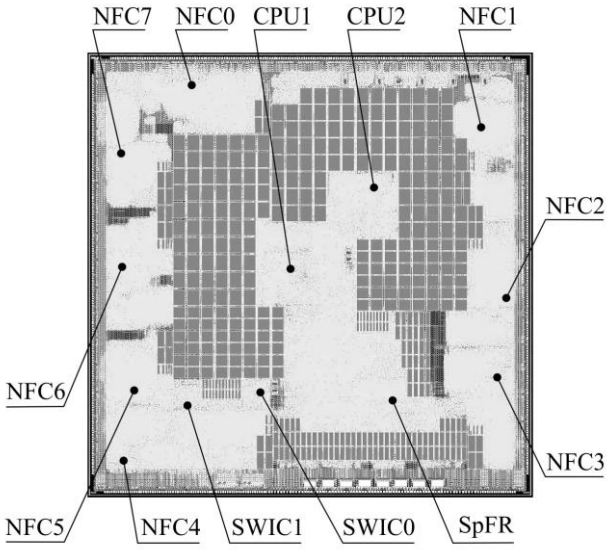


Fig. 3. MCT-04 chip post-layout area. The chip size: 17.5 mm x 17.5 mm.

VI. APPLICATION OF MCT-04

In Fig.4 shown the block diagram of on-board Mass Storage Device.

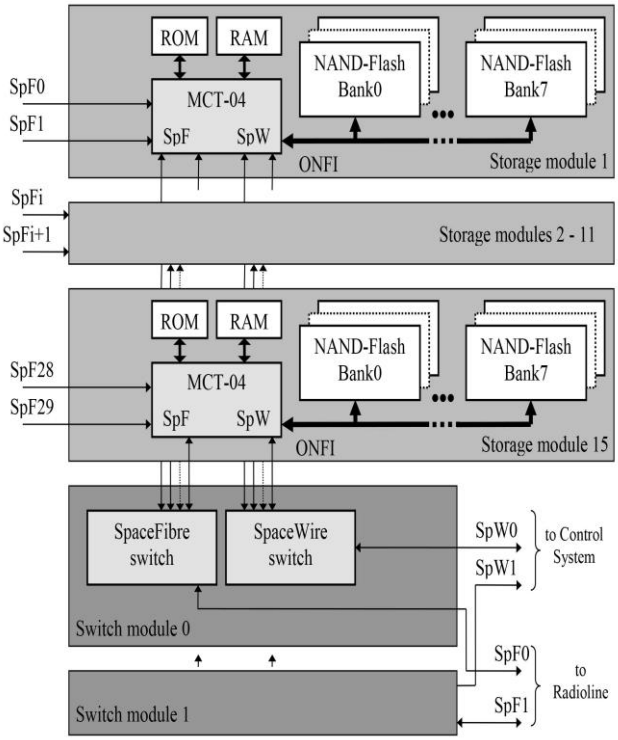


Fig. 4. Block diagram of on-board Mass Storage Device (MSD)

The structure of the on-board MSD suggests applying of SpaceWire and SpaceFibre high-speed interfaces as a communication transmission medium of instructions and data. The SpaceFibre interface will be used for communication of those elements of informational computing system where data transmission rates reach Gb/s per channel. The SpaceWire interface is used as the common unified environment for transmission of commands and interaction between all subsystems of informational computing system.

Structurally the MSD consists of storage modules (from 2 to 15 modules) and two switch modules. Storage modules are intended for reception of input information on two channels of the high-speed SpaceFibre interface and it is saving in NAND-Flash memory. Switch modules are intended for information transfer between storage modules and for formation of an output flow to Earth via a high-speed radiofrequency line.

Basic elements of the storage modules are the MCT-04 (memory Controller) and NAND-Flash memory.

NAND-Flash memory is based on modules available from 3D PLUS [9]. MSD consisting at 15 storage modules has total capacity up to 15 Tbytes.

View of the 3D PLUS module shown in Fig.5.

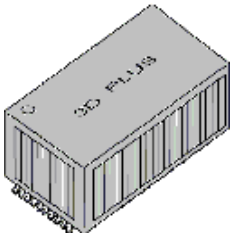


Fig. 5. View of the 3D PLUS NAND Flash storage module

In Fig. 6 shown the block diagram of on-board data processing system.

On-board data processing system consists of Digital Signal Processing System, Instruments and Mass Memory. Digital Signal Processing System is implemented on Radiation tolerant heterogeneous Multi-core ASIC MC-30SF6 [8].

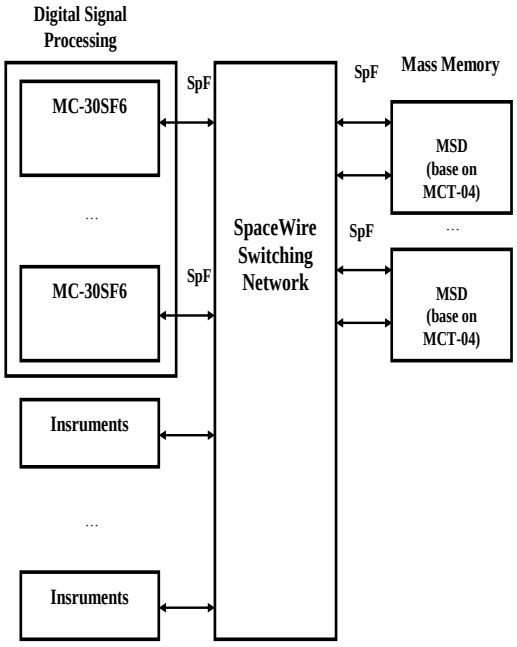


Fig. 6. Block diagram of on-board data. Processing system

VII. CONCLUSION

Radiation tolerant chip MCT- 04 was developed using technology SpaceFibre/GigaSpaceWire (SpaceWire-RUS standard) and is designed to create a network drive terabyte capacity.

Under the ELVEES development is the 90 nm SOC design for the on-board Mass Storage Device with the transmission rates of serial I/O duplex SpaceFibre GigaSpaceWire (SpaceWire-RUS standard), transceivers up to 12 Gb/s per channel. It is planned to provide an opportunity to work via fiber-optic transceivers. Further extension the volume of storage device up to 4 Tbytes is under consideration.

REFERENCES

[1] Next Generation Processor for On-board Payload Data Processing Application ESA Round Table Synthesis, ESA, TEC-EDP/2007.35/RT, October 2007

[2] S.M. Parkes, C. McClements, M. Dunstan and M. Suess, “SpaceFibre: Gbit/s Links For Use On board Spacecraft”, International Astronautically Congress, Daejeon, Korea, 2009, paper IAC-09-B2.5.8

[3] “D2.1 - SpaceWire-RT Outline Specification”, SPACEWIRE-RT Consortium, 06.09.2012.

[4] “D5.1 - SpaceWire-RT ASIC Implementation Feasibility Summary Report”, SPACEWIRE-RT Consortium, 09.05.2013

[5] Requirements-driven Verification Methodology (for Standards Compliance). www.accellera.orq

[6] E. Golovina, M. Makeeve, A. Nikolaev, F. Putrya, A. Smirnov, REUSABLE COMPLEX SOC LEVEL TESTS CREATING AND DEBUGGING METHOD, Problems of Advanced Micro- and Nanoelectronics System Development, N.2, 2015, p. 45-50

[7] F. Putrya, Method of free C++ code migration between SoC level tests and standalone IP-Core UVM environments, Design & Test Symposium (EWDTS), 2014 East-West, p. 1-4

[8] Radiation tolerant heterogeneous Multicore "system on chip" with built-in multichannel SpaceFibre switch for the “intelligent” signals and images processing systems

[9] 3D PLUS, “Radiation Tolerant Memory, FLASH NAND Product Overview”, http://www.3d-plus.com/product.php?type-=1&fm=20

Enabling Advanced Missions on Small Platforms through Designing Cost Effective SpaceWire-based Avionics Solutions in the CubeSat Form Factor

SpaceWire Missions and Applications, Short Paper

Dan Ohlsson, Henrik Löfgren, Emil Vinterhav, Stefan Strålsjö
ÅAC Microtec AB
Uppsala, Sweden
dan.ohlsson@aacmicrotec.com

***Abstract*—When developing the SpaceWire based Sirius Data Handling System (DHS) for small satellites ÅAC Microtec has had to address the dual problem of designing for a low cost and in a small form factor while retaining the high performance, availability and longevity required to support advanced science. In order to achieve the low cost a number of strategies have been applied. Notably, the data handling units are generically designed to be reusable either used for other missions as a part of the platforms they were initially targeted for or for use on other platforms, including CubeSats. Also, COTS components known to withstand the space environment have been used for the designs, in a strategy to reduce component costs. To increase reliability, the Sirius DHS units are designed around a Flash-based FPGA to be tolerant to SEE through triple modular redundancy (TMR) and other fault handling techniques and implements SpaceWire as the main data bus, using nano D-sub connectors to save space and weight. The paper explores in more detail the choices made on the data handling units in order to achieve sufficiently high performance and high reliability in a CubeSat compatible form factor at a reasonable price.**

***Index Terms*—Data Handling System, SpaceWire, microsatellites, COTS.**

I. Introduction

In 2014 ÅAC Microtec was awarded with a project to design a small, low-cost Data Handling System (DHS) for the InnoSat satellite bus, that was to be used in the national science mission MATS [1]. The project was seen by ÅAC Microtec as a possibility to use the experience gained from designing previous DHS systems to make a new design that would fit the new requirements. Designing a more generic DHS would enable a drastic reduction in project cost since parts of the non-recurring engineering (NRE) could then be split over several current and future satellites. This would be a first step to enable the cost reduction required for the project, compared to using traditional space systems.

Figure 1. Prototype of a DHS unit.

The DHS is intended for all data handling between the radio and payload, as well as controlling the power and AOCS systems. Included functions shall be data storage, scheduling of experiments and maneuvers as well as handling the radio, converting payload data to CCSDS (The Consultative Committee for Space Data Systems communications encoding standard) compatible messages that can be sent to the radio.

The MATS science mission is designated for Low Earth Orbit (LEO) and a 2 years lifespan. The requirements were optimized for this, with a starting point in the ECSS (European Cooperation for Space Standardization) standards but in many cases relaxed or pointing to the established industrial standards, giving some freedom to the designers. However, a lot of requirements were still given on the overall reliability of the system, also under radiation, making the alternative of a commercially available computer extremely difficult. The alternative of using existing CubeSat [2] electronics was also considered risky, considering the fact that, despite in some cases having many successful recorded flights, the systems are not well tested for the space environment. Also, a lot of commercial electronics, that can be sensitive to radiation without the proper mitigation techniques, are used in most of those designs.

II. Design philosophy

A. Requirements

The requirements for the DHS were, except for the national science mission, taken from several systems that ÅAC Microtec have previously been and are currently being involved in the design of. Input was also taken from market studies, such as [3], to improve the reusability of the system. In this way a design can be achieved that will fit the ongoing projects at ÅAC Microtec as well as be a more generic system for small satellites.

The size of the satellites being developed at ÅAC Microtec and their partners varies from CubeSats (3U) up to platforms in the range of a cubic meter and 250 kg. This required the DHS to have some kind of scalability but also to be small and lightweight in its most basic forms. Given the development of the market and the experience from the previous systems, those factors were considered important also from the perspective that the DHS was to fit upcoming satellites.

B. System level decisions

To match the different requirements on size, interfaces and computational power, a modular approach was selected. In this way one or a few units could make a complete lightweight DHS for simpler systems while more units could be added to handle the requirements of more complex systems. To handle all the functions required from the DHS, two types of units were identified, one combined mass-memory and radio interface and one more generic unit, designed to act as on board computer and subsystem handler. SpaceWire was selected as internal communication interface, to match the requirements in bitrate and stability as well as enabling connections to external units over a standardized and well known interface. However, using the standard micro-D connectors would significantly increase the unit size, just as would using the quite stiff standard harness. Therefore an approach using nano-D connectors and removing the outer shield and jacket, based on the suggestions in [4] was selected.

What could also be seen from the requirements on the systems was that the cost level had to be kept quite low while the reliability of the system still needed to be high, reducing risks and downtime. Given that the number of units was expected to be relatively large, an approach where unit cost was prioritized compared to NRE was implemented. This led to the conclusion that normal space grade components could not be used other than where no other alternative could be found. Instead commercial off the shelf (COTS) components were to be used. In our case COTS is referring to components that have not been specifically designed or qualified for use in space.

To maintain the reliability of the design, several mitigation techniques as well as thorough testing was needed. The experience at ÅAC Microtec was that a COTS based design could become very reliable and definitely sufficient for the purpose, even though the total reliability would not be as high and well defined as it would have been using space-grade components,

The parts of the system that were considered the most sensitive to single event effects (SEE) are the processor and memories. Instead of using sensitive commercial processors, and to be able to better monitor the memories, a solution based on a flash-based FPGA (field programmable gate array) with a dedicated system on chip (SoC), including a soft processor, was selected. This also enabled better tailoring of the IO capabilities, like including CCSDS encoding and decoding directly in hardware, offloading the processor.

Even using the COTS approach would prove too costly for the intended range if each component was to be qualified individually. Therefore a system-level test approach for testing was selected, which would still give a good proof of the reliability of the system while the exact properties of each component would be more difficult to obtain. To reduce the risk during testing and flight, careful component selection was needed.

III. COTS versus space grade

Building systems for space using COTS components adds a number of difficulties to overcome, amongst others that the components are:

- Not tested for the space environment
- Designed with restrictions that are not applicable to usage in space
- Manufactured using cost efficient methods that are not adapted for usage in space
- Supported by suppliers that lack the knowledge and understanding of using the components in space systems
- Not unit or batch tested to the same extent as space grade components

The most obvious difficulty is that the COTS components have not been tested for use in the space environment, leading to undefined behavior, especially when looking at radiation effects. This increases the requirements on component selection and testing procedures as well as forcing measures to be taken to reduce the impact of any effects that might still occur.

Requirements that are not valid for space have often been imposed, whereof RoHS (the Restriction of Hazardous Substances Directive), resulting in the usage of leadless solder, is one of the more troublesome. In smaller satellites the thermal mass is quite low and this, together with the fact that less resources is available for active thermal handling, increases the temperature swings that can be expected. Leaded solder is generally better in handling thermal stress than the leadless solders often used in the industry [5]. Also, while the ground-based industry is taking the problem into account and has made improvements [6] the fact still remains that solders with a very high amount of pure tin can have problems with the formation of tin whiskers. While components with leaded terminations can still be found, this was considered very limiting for the component selection. Also, retermination can be made, but this would add an extra process step increasing cost, risks and lead time. The mitigation techniques used to cope with the problems of unleaded solder are described in section IV.

While space and military grade components have mainly stayed with ceramic and metal packages, the commercial industry have moved to plastic packages, mainly to reduce cost but also to decrease weight and vibration sensitivity. When plastic packages were first introduced the differences in reliability compared to ceramic packages were large, but since then the plastic packages have improved significantly and problems such as sensitivity to temperature cycling and hermeticity are no longer large concerns. In the data handling system (DHS) no extreme temperatures are expected, so the greater sensitivity to high temperatures of plastic packages is not seen as a problem either.

Another problem in using COTS components is that some parameters that can easily be obtained from space grade component suppliers can prove difficult to obtain from COTS suppliers. This can be outgassing properties, batch tracking data, the performance under thermal stress or mitigation techniques used to reduce the tin whisker problem. While this problem increases costs for non-recurring engineering (NRE) somewhat, it is not a big cost driver and does not affect reliability.

Finally a typical COTS component is not subject to the same rigorous testing on unit and/or batch level as a space grade one. This puts higher requirements on later testing in general and acceptance testing of the units after assembly in particular.

Looking at the other side a few of the advantages are:

- Cost
- Availability
- Basic quality
- Performance

The most obvious advantages of COTS are the cost and availability. Having much higher volumes, plastic packaging and less rigorous unit testing decreases cost significantly, about a factor of 100 per component is not uncommon.

A not so obvious advantage is the basic quality. While space grade components are even more thoroughly tested to find any problems than many COTS components, the later have the advantage of often being produced in huge quantities and in more mainstream processes. That means that even just a few of them are tested per batch the number of batches provide better statistics for improvements in the processes.

Another advantage is performance. Commercial processors and memories are often many years ahead of the space grade counterparts, which can also be seen from the fact that also the traditional space industry in some cases go to the COTS market to enable their missions [7].

In some cases, especially in critical power paths, the component needs to be used on the edge of its capacity and therefore needs to be well defined also during and after radiation. For those rare cases space grade components are used in the systems to fulfill the requirements. No such cases have been identified for the DHS while some have been found in the power subsystems that were designed at the same time.

IV. Mitigation techniques

As described in the previous section, using COTS in space comes with a large number of limitations. To handle this and to make the system robust to faults present also for space-grade based designs, a number of techniques are implemented on different levels of the DHS.

A. System

On a system level housekeeping and power monitoring are the primary ways to detect and handle any failures.

Power monitoring is done both on the different DHS units and on the central power control and distribution unit [8], which is not the focus of this paper. On the DHS side, each unit has a wide input voltage range, to be able to work in many different systems, and use number of protection and mitigation functions:

- Overvoltage protection, shutting down the power input in the case of a malfunction of the power supply
- Undervoltage lockout, ensuring that current leakage does not put the unit in an unknown condition.
- Power loss detection, giving the processor a heads up of a few microseconds for a graceful shutdown.

In general, more resets can be expected from a COTS based than a space grade based solution, but tests performed and flight record have shown that such resets are very rare and generally not related to problems in the power handling. In all, the power monitoring systems implemented are considered enough to avoid most catastrophic failures due to SEE even though the units are not designed to be single point failure-free.

The DHS units provide a number of housekeeping parameters that are used by the system software to decide on actions to prevent or analyze failures. Those also provide feedback on the behavior of the system during testing.

Despite all the measures taken to have reliable and self-healing units there is always a risk of having units that gets locked-up due to untested corner cases in the software implementation. To mitigate this all DHS units are capable of being reprogrammed in flight. This is used together with the possibility of resetting the units to the updated or original software image using hardware decoded CCSDS compatible messages in the ECSS-E70-41A packet utilization standard (PUS) format.

Finally both the original and updated software images are tripled, so if booting from one version of the specific image fails due to single event upsets (SEUs) corrupting the data, the system can move to the next one.

B. System on Chip (SoC)

To provide a system much less sensitive to SEE than using a commercial processor, an FPGA (Field Programmable Gate Array) with a dedicated SoC is used. The FPGA selected is Flash based, which gives its configuration an insensitivity to SEE. Since the configured gates can still be sensitive to SEU:s all flip-flops of the SoC used in space are tripled, using majority voting to decide on the outcome of the operation. Also, all caches in the processor are controlled using parity, provoking a cache reload if any faulty register is detected. This

renders the SoC, and thereby the processor in itself, more or less immune to SEU:s, removing a large risk with a commercial design, still using COTS components.

To mitigate the problem of SEU:s in the memories, reads from all memories are checked using error detection and correction code (EDAC). While the data from the non-volatile memories is just checked when read out, the volatile memories are also continuously scrubbed and any errors found reported and, if possible, corrected. If an uncorrectable error is found during the read from the volatile memory on a command to be executed, a reset is triggered, putting a fresh image from the non-volatile memory into the volatile one.

To prevent any software related issues from locking the system up and to cover for errors not found by the other systems, a watchdog is implemented in the SoC, causing a reboot of the unit if it becomes unresponsive.

Finally, all peripherals in the SoC have parity checking of their FIFO (First In First Out) registers, enabling data to be resent if needed.

Figure 2 gives a rough understanding of the resulting unit. Most parts of the system, including the processor, runs at 50MHz.

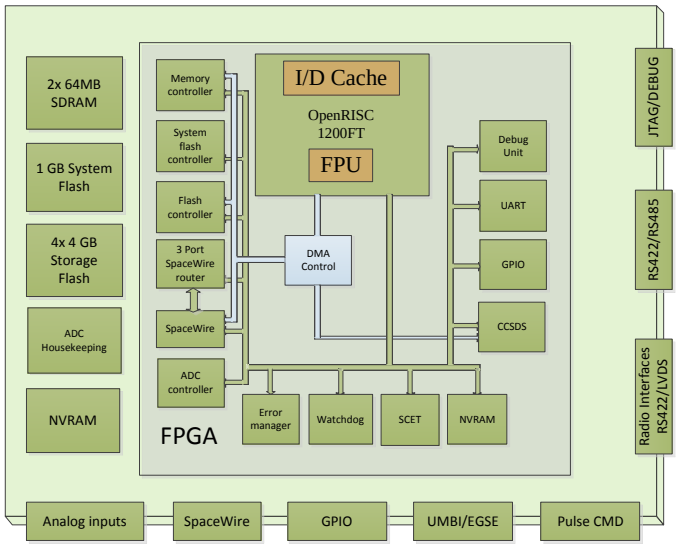


Figure 2. Sample unit block diagram.

C. Electronics design and manufacturing

Using COTS components with unknown reaction to the space environment in the design introduces a risk in the test stage, especially for radiation testing, as well as in the final application. Therefore careful component selection is of uttermost importance to reduce final cost and improve the resulting reliability. A lot of effort has been spent on finding components that are considered to be relatively safe to use. For instance, already published radiation reports on different COTS memories are used to find suitable SDRAM (synchronous dynamic random access memory) and Flash (non-volatile) memories for the DHS. A lot of effort has also been spent finding data that is more relevant to satellite systems than COTS-based systems, such as outgassing properties.

As previously mentioned the DHS is assembled using unleaded solder. The two main risks with this are possible failures due to temperature variations and the risk of tin whiskers.

To reduce the risk of damages to the solder joints due to thermal stress, interfaces known to be sensitive (such as chip scale packages or ball grid arrays) were avoided where possible and tested and analyzed when used.

Tin whiskers are crystalline spikes of tin that grow from tin surfaces and that can create shorts between adjacent conductors or break free and short conductors in other parts of the system. While the effect seems to be impossible to completely avoid for components with terminations plated with pure tin, it can be mitigated using conformal coating that reduces growth and protects all other surfaces from any whiskers that are able to grow through the coating. The coating also helps protecting the electronics from any moisture that it might be exposed to before launch.

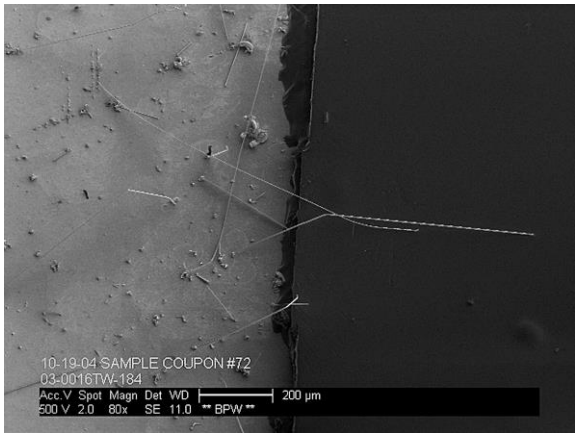


Figure 3. Image showing the formation of tin whiskers on uncoated, to the left, and coated areas, to the right [9].

Since qualification testing (described further in section V) is one of the cost drivers in the development of the DHS it is important to make sure that no changes are done in the manufacturing processes without an analysis of the consequences for the test results. This is controlled using lot travelers where the manufacturer enters the parameters that are used in the different processes. The same lot traveler is also used to increase traceability and ensure that the correct process parameters and safety measures are made.

To simplify manufacturing, and thereby reduce costs, industrial standards rather than the space specific ones have mostly been used to define the quality requirements for those steps. Since only a few manufacturers are used to the space standards, this increases the selection of manufacturers significantly. In general the highest levels of quality were been selected, such as assembly according to IPC-A-610 class 3.

D. Mechanics

The DHS units are also separately enclosed in mechanic boxes, adding to the radiation shielding and insensitivity to electrostatic discharges during assembly. This also helps in harnessing, providing fastening points for the cables and simplifies integration.

V. Testing

COTS components are not uncommon in space designs, also in designs with even higher requirements on reliability than the ÅAC Systems, but there are few, if any, standards for space systems that are adapted to anything else than the very highest (and therefore very expensive) quality level. Since ÅAC Microtec depends on external resources doing most of the environmental testing (except for temperature and humidity), it is also important that any test specifications used are either accepted and known or, for less complex tests, clear enough to be able to do the testing without following an existing standard. To match the requirements seen an internal standard have had to be made, being a compromise between the ECSS standards, NASA recommendations and the cost efficiency required. In some cases industrial or military standards were selected to increase the number of available test facilities, decreasing cost and potentially lead times. As an example, approach very close to the ECSS standard was used for the temperature cycling while the SEE testing was done with protons only, based on NASA recommendations, to enable unit testing and the usage of nearby facilities.

A problem in defining the test levels have been that the ECSS standard is mainly adapted for testing for a specific mission, where a lot of parameters are given by the selection of orbit and launcher. Since the DHS was designed to be reused in future missions, a more generic approach have had to be selected, where some test levels have been taken from the ECSS or industrial standards while others have been tailored to get the most reliability for a given cost. This leaves the decision of any additional testing with the customer, if the test levels presented are not enough.

VI. Conclusion

A. Summary

Using in-house knowledge from fully space grade and purely industrial designs, a design and test philosophy matching the requirements for the mission and more general requirements, improving the reusability of the system, has been made. Those incorporate ECSS, NASA as well as established industrial standards and recommendations, reducing the cost while only slightly decreasing the overall reliability compared to a full ECSS test campaign. Experience from previous test campaigns and flights have been incorporated to reduce the risks and increase the reliability of the system substantially. This results in a system that is drastically less expensive than a fully ECSS compliant system but that still have a lot of the reliability that can be expected of such a system for the intended usage, which is 5 years in LEO.

B. Future work

A first version of the DHS is due for launch on a tech demonstrator satellite in 2016 and the data obtained from the system during flight will be analysed and used for improvements.

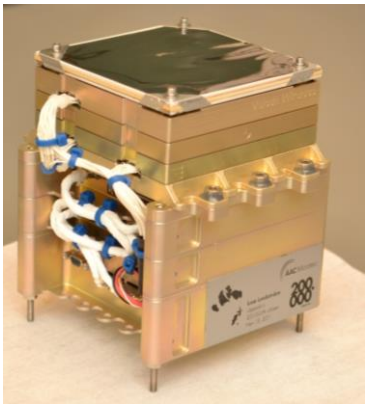


Figure 4. Integrated DHS with radio and power unit attached.

A continuous process of improving the design and test philosophies and making the units even better adapted for reusability is ongoing. In this process there is a continued weighting of the cost versus the reliability in a way that a reliability good enough for most LEO small satellite missions, including the more critical ones, is achieved, without getting the costs associated with electronics for a large satellite designed for long time usage in higher orbits.

References

[1] N. Larsson et al," InnoSat and MATS – An Ingenious Spacecraft Platform applied to Mesospheric Tomography and Spectroscopy" 10th IAA Symposium for Earth Observation: 2015

[2] Mehrparvar, Arash, "CubeSat Design Specification" The CubeSat Program, CalPoly SLO. Retrieved September 2015

[3] Doncaster B and Shulman J, "2016 Nano/Microsatellite Market Forecast", SpaceWorks Enterprises, INC.

[4] Rouchaud G., Ilstad J. and Mettendorff F., "LOW MASS SPACEWIRE", International SpaceWire Conference 2011

[5] Garcia C, "NASA-DoD COMBINED ENVIRONMENTS TESTING RESULTS", SMTA International conference 2010

[6] Collins P, "Lead Free Plating & Soldering", Harwin plc

[7] Dellandrea B., Estaves G. and Alison B., "System Definition of COTS-based computer for on-board systems", ESA TEC-SW & TEC-ED final presentation days: 9 th December 2015.

[8] Tsamsakizoglou M., Löfgren H. and Gunnarsson M., "MICROSATELLITE POWER CONTROL AND DISTRIBUTION UNIT FOR THE INNOSAT PLATFORM", 11th European Space Power Conference, Thessaloniki, GR, 3-7 October 2016

[9] Woodrow T.A and Ledbury E.A., "Evaluation of Conformal Coatings as a Tin Whisker Mitigation Strategy", IPC/JEDEC 8th International Conference on Lead-Free Electronic Components and Assemblies, San Jose, CA, April 18-20, 2005.

Thursday 27 October

Components (Long)

Compact, Impedance-matched SpaceWire Connector Development – "MicroMach SpaceWire"

SpaceWire Components, Long Paper

Kevin Enouf
Axocom Space Products Division
Axon' Cable SAS
Montmirail, France
k.enouf@axon-cable.com

Stéphane Hermant
Axocom Space Products Division
Axon' Cable SAS
Montmirail, France
s.hermant@axon-cable.com

***Index Terms*—Connector new design, high data rate (HDR).**

I. Introduction

Axon' Cable was selected to carry out the development of a new, compact impedance-matched SpaceWire (abbreviated herein to "SpW") connector under an ESA Technology Research Project. The classic existing SpW connector, the 9 way micro-D, whilst having the advantages of being both small and common to many projects, is not electrically optimized to the SpW needs, nor is it particularly efficient when it comes to EMC protection. As high data rate applications are booming, there is, a pressing need to develop an improved connector interface.

The design phase of the project under ESA supervision is nearing completion. In one hand Axon' has carried out a user survey to assess the needs from users and their priorities for such a connector. In the other hand Axon' has also conducted a connector survey to identify and evaluate potential candidates already on the high data rate connector market from different manufacturers.

From the original scope of the project, Axon' has already gone the extra mile with the design and manufacture of a number of different cable constructions in order to evaluate whether the existing four shielded twisted pairs configuration is indeed the best option for SpW L.V.D.S. transmission.

Axon' will present the main results and conclusions of these surveys, along with a detailed presentation of the prototype of the compact, impedance-matched connector, and a description of the trade-off made in order to achieve a desirable size coupled with significantly improved electrical and EMC performances.

Note: To avoid repeating the lengthy phrase "compact impedance-matched SpaceWire connector" throughout this paper and elsewhere, Axon' has adopted the working name "MicroMach SpW" for this connector family, drawing on its twin heritages of micro-D and AxoMach® ('mach' meaning high speed) technologies.

II. Surveys and Characterization

A. Existing Market Connectors Survey

Among a variety of existing connectors on the market dedicated to interconnect high speed links, only a few appear to meet the required electrical performance levels, but those tend to be much larger in size than the 9 way micro-D solution (see example in fig.1). Other than its compact size, however, the micro-D offers the least electrically compliant results of the study - unsurprising in that it was originally chosen for its size and robustness rather than its HDR capability. Put another way, typically when a connector is of a desirable size, the electrical parameters tend to be compromised - particularly in terms of EMC performance. Additionally, the various available connectors on the market, along with their contacts and accessories, are not always well matched to the cable size and can therefore create a degree of electrical mismatching, generally manifested by deviations in characteristic impedance and shielding efficiency.

Fig.1. Size comparison between classic 9 way micro-D connector and one of the top HDR performers in the connector survey

As no connector on the market currently meets both the electrical performances and the dimensional aspirations of the study, the development of a dedicated connector was commenced, focusing on the twin targets of remaining as close as possible to the size of the 9 way micro-D whilst significantly improving the overall electrical performances.

B. Cable survey and trials: first results

A cable survey was carried out by Axon' Cable to try to improve cable features based on skew reduction and size. The approach has been to consider a manufacturing process of shielded parallel pairs using the same low loss A-PTFE® dielectric material as used in the current low mass SpW cable ESCC3902/004. The intention is to scale up the data rate capability of a shielded parallel pair cable whilst ensuring an intra pair skew and insertion losses reduction.

With shielded parallel pairs, the skew is better managed due to low variation of the wires length along the cable (a twist adds an extra length per wire compared to a parallel pair). This parallel wire layout leads to a fair reduction of the insertion losses of the cable. Moreover from a mechanical point of view, the overall diameter of the wires assembly is smaller in a parallel construction than in a twisted one because no filler material is required between wires.

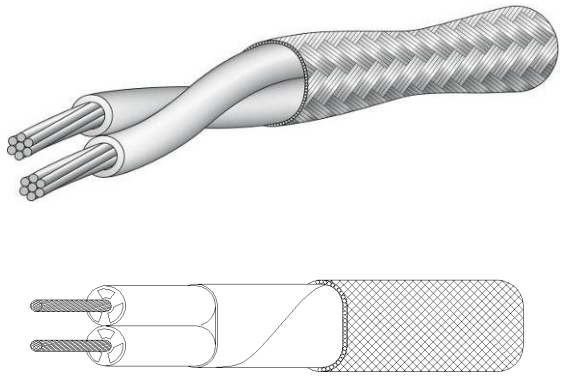


Fig.2. Shielded twisted pair (top) versus shielded parallel pair (bottom)

In an attempt to reduce the cable dimensions even further, a shielded twisted quad cable configuration (fig.3) has been investigated. The wire baseline is equivalent to parallel or twisted pair but matched to offer 100 Ohms differential impedance between opposite wires. This cable requires a dedicated connector with four pins within the same contact (quadrax) to achieve the required electrical performances.

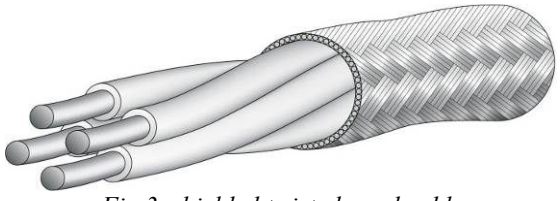


Fig.3. shielded twisted quad cable

C. Electrical characterization test and report

Electrical features both for time domain and frequency domain have been selected with ESA to characterize the future SpW connector and link. Based on this selection, Axon' has built prototype assemblies using existing high data rate connectors and its low mass SpW cable. Crucial electrical parameters like scattering parameters, characteristic impedance and crosstalk have been measured as well as new parameters, such as differential mode to common mode conversion, characterizing the symmetry of the differential transmission line.

III. NEW CONNECTOR DESIGN

The shape of the proposed new connector was rapidly chosen in accordance with customers' needs to be a rectangular design with four separate cavities. Each cavity is separated by a metallic wall to improve crosstalk performance. The four ways are designed to all be fully 100 Ω adapted throughout the complete transmission line.

To secure the mating sequence, two special guide pins are used which, as well as securing the backshell to the connector, help accurately guide the male and female connectors together during the mating operation.

The electrical contacts are assured by the very well-known and reliable Twist Pin technology used on micro-D connectors, which can boast decades of successful flight heritage. These contacts are inserted by first fitting them into dielectrics which are then press-fitted into the connector shell. This design prevents the contacts moving backwards or forwards within the connector.

A SpW cable consists of four inner shields (around the twisted pairs) and one overall shield. One of the main challenges of this new development, therefore, was to design a connector with four effective inner shield terminations in an overall size as close as possible to that of a 9 way micro-D. The choice, made jointly with ESA and STAR-Dundee, was to work on a connector with "good-but-not-360°" inner shield termination (as illustrated in fig.4) in order to make it more compact. The purpose of this design is to guarantee sufficient electrical contact between the braided shield of each pair and the metallic shell of the connector whilst saving space and significantly reducing crosstalk.

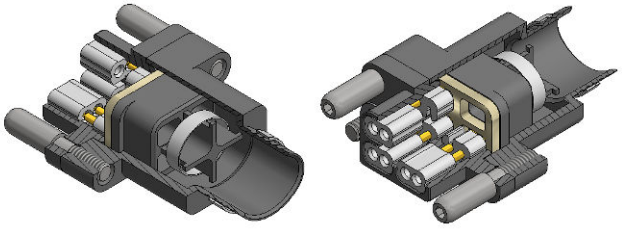


Fig.4. 3D cross-section of the in-line male MicroMach SpW connector

The contact of all four inner shields is achieved using a metallic 'nano' band tightened around a special feedthrough insert with the four shielded pair in situ (fig.5). The cruciform shape at the rear of this inner shield insert ensures a solid electrical contact by maintaining a degree of pressure over the 4 cable braids. This Axon-designed insert has been dubbed internally, "aXiform"

The overall (outer) shield of the cable is then crimped over the backshell funnel with an axoclamp® (or equivalent) banding adaptor.

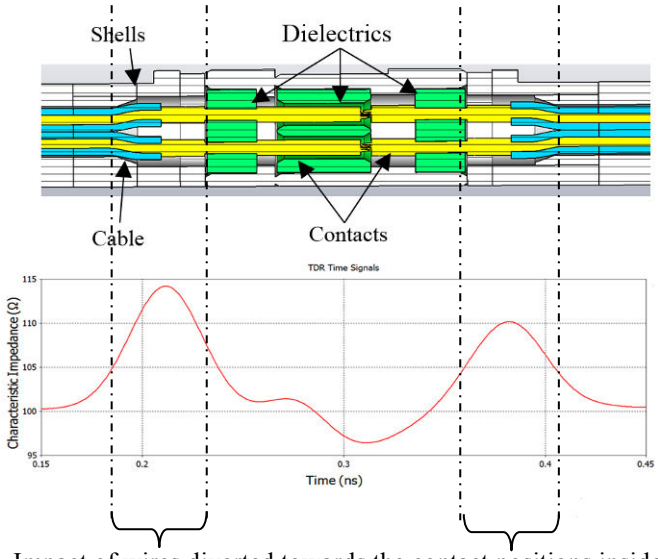
Fig.5. twisted pair shield connection demonstration with "aXiform" inner shield feedthrough insert

The MicroMach SpW connector is currently designed for both AWG26 and AWG28 SpW cable variants with a specific "aXiform" insert for each size. Other cable constructions could be achieved simply by adapting the insert as required.

Finite Element Simulation

To identify the best compromise between the hardware design and the resulting electrical performance, Axon' carried out Finite Element simulation on 3D models using CST software. These analyses were principally focused on characteristic impedance (Zc) in order to determine the optimum size of all the inner connector elements.

As can be seen in fig. 6, the main mismatching is where the cable is terminated to the contacts (peaks of Zc). Just before the crimped contact interface the Zc variation may be around 20 Ω for AWG28 and 15 Ω for AWG26 cable.



Impact of wires diverted towards the contact positions inside the connector

Fig.6. CST simulation with Finite Element Model

IV. PROTOTYPING

A. Assembly steps description

A cross-section view of the prototype CAD model (fig.7) illustrates the final product showing its different piece parts put together.

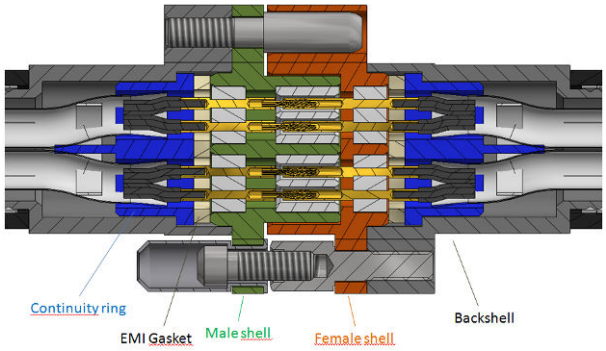
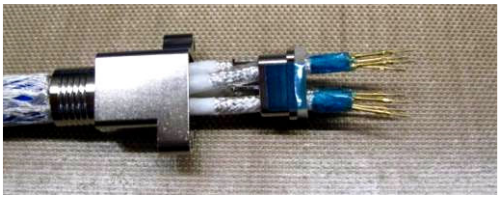


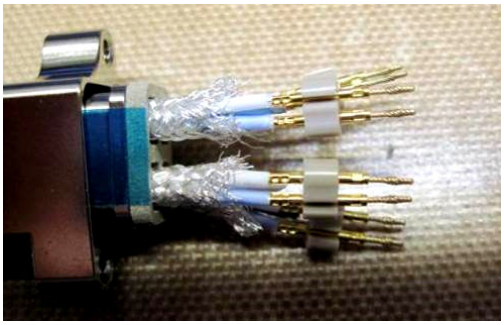
Fig.7. Cross-section view of mated connectors

The next page shows the main stages of the manufacturing sequence carried out by Axon' Cable on a prototype to link cable with newly designed connector and accessories.

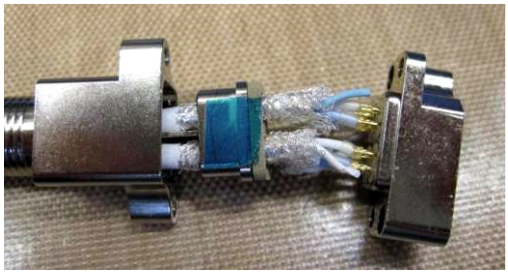
Stage 1: Cable cut to length and wires end stripped. Crimped contact to wires and use of makeshift inner shields protection with blue tape.



Stage 2: Crimped wires insertion (from left to right in the picture) through metallic backshell, continuity ring and through EMI gasket.



Stage 3: Contacts insertion into the flexible plastic dielectric (press-fit)



Stage 4: Dielectrics insertion into the connector shell (press-fit)

Stage 5: Final connector assembly and connection of the external braided shield of the cable to backshell by clamping

B. MicroMach SpW main physical properties

The connector flange (rectangular) dimensions are 21.4 x 9.3 mm. The largest cross-section is 32% bigger than the existing 9 way micro-D but a comparative view showing side-by-side connectors in fig.8 indicates how similar size-wise the connectors really are.

The mass of the MicroMach SpW female connector including backshell and screwlocks is only 6 grams and the male version mass sits at 7.5 grams. The masses have been estimated from CAD models and need to be confirmed at the end of the project.

Fig.8. 9 way micro-D connector (left) and MicroMach SpW connector (right)

C. Preliminary electrical measurements

Static measurements:

Contact bonding resistance (between male & female) < 5 mΩ.

Frequency domain:

Crosstalk Next/Fext < -50 dB up to 1 GHz.

Return loss < -20 dB up to 1 GHz.

Time domain:

The following characteristic impedance Zc (fig.9) is measured on a 9 way micro-D connector couple (male and

female) connected to ESCC 3902.004.01 low mass SpW cable.

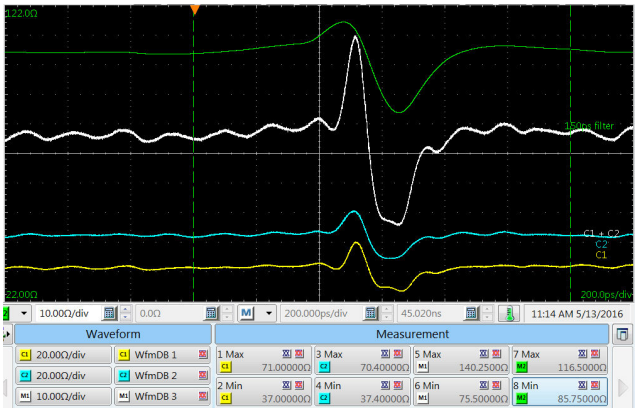


Fig.9. Zc on 9 way micro-D connector

Zc varies between 86 and 116 Ω (with 150 ps rise time filter, green plot) & from 76 to 140 Ω (in full band, white plot)

The following characteristic impedance (fig.10) is measured on a MicroMach SpW connector couple (male and female) connected to ESCC 3902.003.02 SpW cable

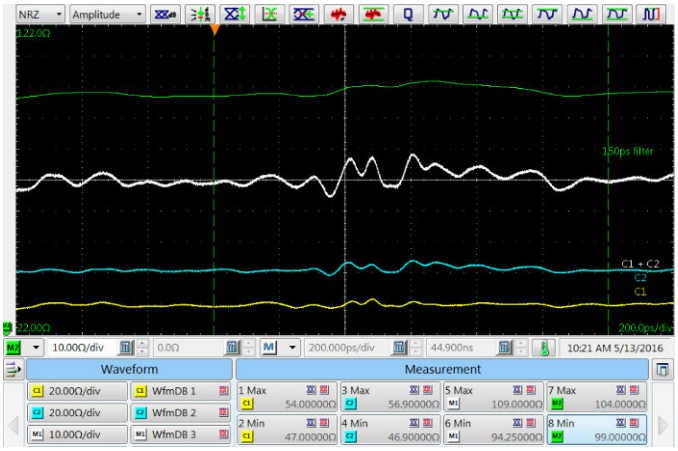


Fig.10. Zc on MicroMach SpW connector

Zc varies between 99 and 104 Ω (with 150pS rise time filter, green plot) & from 94 to 109 Ω (in full band, white plot).

Eye pattern / SpW mask compliancy:

The following eye pattern (fig.11) is measured on a 1-metre link using ESCC 3902.004.01 low mass SpW cable connected to 2 connector couples and run at 4 Gb/s.

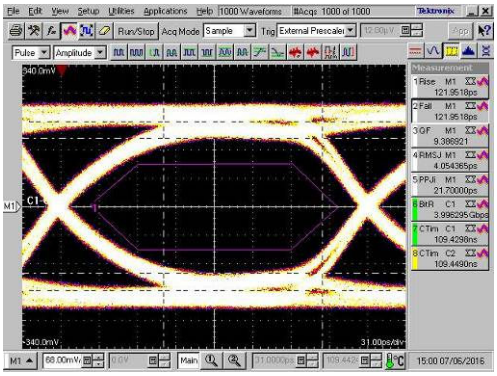


Fig.11. Eye pattern test result (mask with purple colour)

Fig.11 shows compliancy of the vehicle under test to the requirement (measured eye remains outside the mask representation).

D. Presentation of MicroMach SpW Range

Axon' has also worked on a number of different possible connector variants as presented briefly below. Some additional PCB connectors will be developed according to the need.

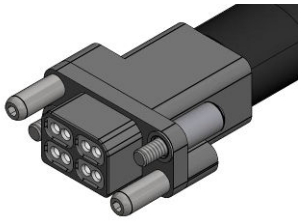


Fig.12. Inline Male

This connector (fig.12) will be used mainly for normal links between various equipment or between equipment and router.

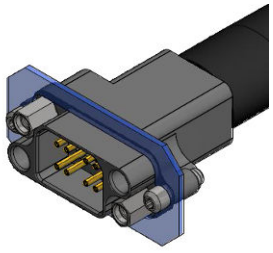


Fig.13. Inline panel mount

This variant (fig.13) will be used to add a break point in a link. It could be fixed to a dedicated bracket or on a panel.

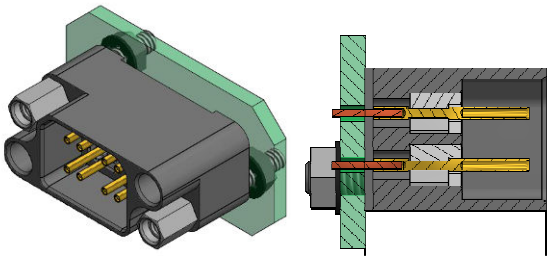


Fig.14. Board Straight PCB

Basic PCB connector (fig.14) can be used to connect to a board with limited mismatching and crosstalk.

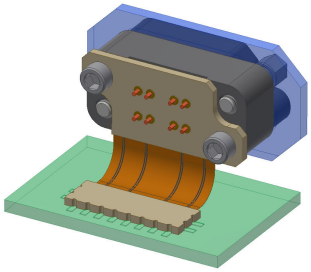


Fig.15. Flex PCB panel mount

This connector assembly (fig.15) allows a good mechanical decoupling between PCB and equipment panel while maintaining impedance matching and crosstalk reduction. The skew is also very low.

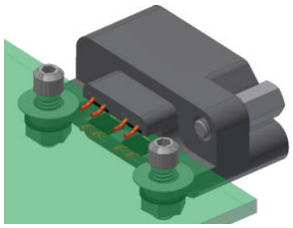


Fig.16. Edge PCB SMT

This variant (fig.16) saves a lot of space on the PCB and allows a significant crosstalk reduction between the two connection sides.

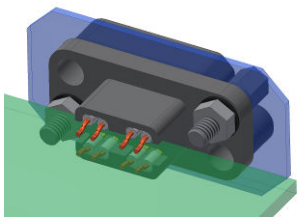


Fig.17. Edge PCB SMT panel mount

The connector (fig.17) adds the possibility of mounting the connector on a panel (rear mount)

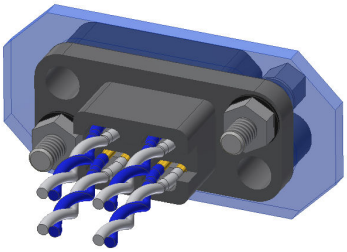


Fig.18. Wired PCB panel mount

This variant (fig.18) allows a panel mount while maintaining impedance matching. Offering easy to solder connection to PCB.

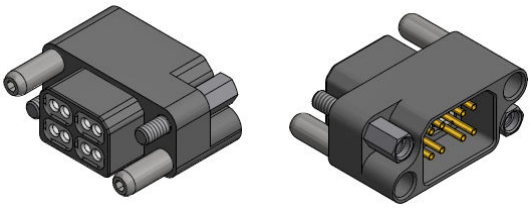


Fig.19. Saver (front and rear view)

Savers (fig.19) are often needed during the Assembly, Integration and Test phase.

V. CONCLUSION

Trade-off had to be made with regards to the cable shield termination to the connector backshell (not full 360° screening) to minimize the size. But the new MicroMach SpW connector range, planned to be available by end 2017 offers, in a size only slightly larger than the current 9 way micro-D, significantly improved performances in data rate, EMC and crosstalk compared to any of the current market solutions.

The Axon' internal cable survey, still in progress, will also propose new possibilities for cabling. Parallel pairs are already showing promising results and could be used with the MicroMach SpW connector to achieve high data rates where other restrictions, such as limited available space, may apply.

An important outcome of this project will be the creation of a generic harness specification for high data rate links plus a new ESCC detail specification characterizing these new SpW in-line and PCB connectors. At the same time the latest revision of the SpW standard, ECSS-Q-ST-50-12 has been issued incorporating the possibility to use this new "MicroMach SpW" connector as a type B.

Radiation-Tolerant 18x SpaceWire Router Design and Qualification for space application – GR718B

Components, Long Paper

Sandi Habinc, Fredrik Johansson, Francisco Hernandez, Fredrik Sturesson, Felix Siege
Cobham Gaisler
Kungsgatan 12, SE-411 91, Göteborg, Sweden
info@gaisler.com

Martin Suess, Rok Dittrich
European Space Agency
Keplerlaan 1, PO box 299, NL-2220AG Noordwjik,
The Netherlands
martin.suess@esa.int
rok.dittrich@esa.int

***Abstract*— The GR718B is a standalone, radiation tolerant, 18x-port SpaceWire router, enhanced with support for 64 interrupts, SpaceWire-D and SpaceWire standard revision 1. A brief overview of the design background of this device is provided herein. The main design features of the GR718B are also described in this paper. A short summary of the screening and qualification flows for space qualification of the GR718B is given herein. GR718B prototypes have already passed functional and electrical verification and the first flight lot of GR718B is now in its final stage of space qualification. The main outcome the Single Event Effects testing of the GR718B are given here and support the choice of radiation hardening features selected for the design of this device.**

***Index Terms*—SpaceWire, SpaceWire standard revision 1, SpaceWire-D, Networking, Spacecraft Electronics, Router, Radiation, Single event effects,**

I. Introduction

GR718B is a radiation tolerant 18 port standalone SpaceWire router component developed by Cobham Gaisler AB in an activity initiated by the European Space Agency (ESA). The GR718B is currently being qualified for space following an ESCC9000 lot validation approach. The GR718B SpaceWire router has been updated to the latest SpaceWire standard for 64 interrupts codes. The GR718B SpaceWire router use on-chip LVDS (Low Voltage Differential Signaling) transceivers and LVTLL (Low voltage Transistor-Transistor Logic) ports, which have been proven to be operational above 200 Mbit/s. UART and JTAG interfaces, that give access to the on-chip AMBA AHB bus, are provided for configuration and debugging. SPI and GPIO interfaces are accessible through the configuration port, which allows SPI devices to be accessed and general purpose signaling to be performed through RMAP commands. In addition to the mandatory features in the current ECSS SpaceWire standard, GR718B supports group adaptive routing for path addresses and packet distribution. It also includes support for the SpaceWire standard revision 1 (ECSS-E-ST-50-12C Rev.1), the SpaceWire-D protocol, and the updated SpaceWire Plug-and-Play protocol.

The technology selected for the manufacturing of the GR718B is CMOS 180nm, using the DARE+ library from imec(Belgium). The GR718B is currently provided in a 256 pin CQFP package. A prototype board for evaluation and software development for the GR718B has been designed and manufactured. This board can be ordered directly from Cobham Gaisler [1].

Fig. 1. *GR718 device mounted on printed circuit board (PCB)*

II. *Background*

Both ESA and several companies in the space industry have indicated 16 as the most viable number of Space-Wire ports for routers in the near future. Cobham Gaisler's intentions with the GR718B development was to provide this key component. The design is based on the GRSPWROUTER configurable SpaceWire router IP core. The IP core supports from 2 to 31 ports of three different types: SpaceWire, AMBA and FIFO. The SpaceWire ports implement an encoder-decoder compliant to ECSS-E-ST-50-12C [2] and provides an external SpaceWire interface. FIFO ports provide 9-bit parallel interfaces with control signals in each direction (read/write), which can be

used to interface external units or to cascade two or more routers without any glue logic. The AMBA ports interface to an AMBA AHB bus using DMA on the bus. All three port types connect to the switch matrix of the IP core using identical FIFO based interfaces. There is no way to distinguish the three ports on the SpaceWire packet level and upwards. The configurability provided by the IP core makes it usable in many different applications. It has already been used in several standard radiation-hardened components such as Actel's RTAX2000SL and RTProASIC3 FPGAs, and is also used in the Next Generation Micro Processor, the GR740 [3], system-on-chip activity funded by the European Space Agency.

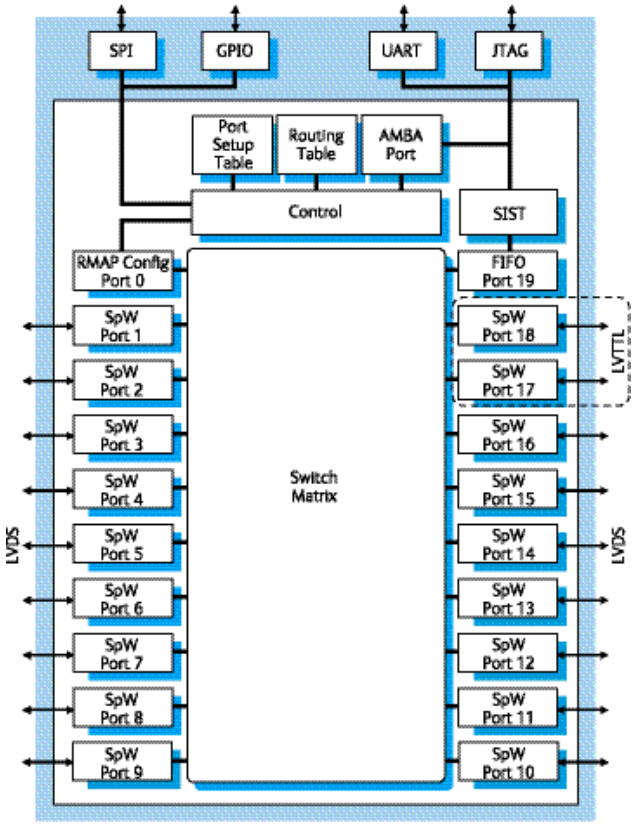


Fig. 2. GR718B architecture overview

During the development phase, two configurations of the IP core were identified as potential candidates for the final ASIC: one with 16 SpaceWire ports with on-chip LVDS transceivers, and two additional ports, either SpaceWire LVTTL ports or FIFO ports; and the other with 16 SpaceWire ports and two internal AMBA ports connected to a PCI interface. Both configurations were evaluated in detail to determine which one would eventually be used for manufacturing. The final choice fell on the configuration with 16 LVDS SpaceWire ports and two LVTTL SpaceWire ports, where the only difference between the two different SpaceWire port types is the I/O type of the pads.

III. Functional Overview

The full GR718B architecture, shown in Figure 2, includes the following modules: SpaceWire Router, SPI Controller, UART Interface, JTAG Interface, General Purpose I/O Interface, SpaceWire In-System Test (SIST), System Level Test Configuration, AMBA AHB controller and AMBA APB controller.

The SpaceWire router implements a SpaceWire routing switch as defined in ECSS-E-ST-50-12C. Among the features supported by the router are: group adaptive routing, packet distribution, system time-distribution, distributed interrupts, port timers to recover from deadlock situations, and SpaceWire-D packet truncation based time-slot violations.

A total of 20 ports is provided, where port 0 is the mandatory configuration port, ports 1-18 are SpaceWire ports, and port 19 is a custom port called the SIST port. Each SpaceWire port contains a SpaceWire codec, and provides an external SpaceWire interface. The SIST port provides a FIFO interface which is internally connected to a SpaceWire In-System Test module (described later). The configuration port provides a target for the Remote Memory Access Protocol (RMAP) defined by ECSS-E-ST-50-52C [4], and an AMBA AHB slave interface, both used for accessing internal configuration and status registers. The configuration port also provides a SpaceWire Plug-and-Play interface, allowing device identification. The ports which are allowed for configuration access can be restricted if needed using several configuration options.

For diagnostic and test purposes, UART and JTAG interfaces are provided. These low pin count interfaces are suitable for small packages but at the same time have sufficient bandwidth. Both the UART and JTAG interfaces act as masters on the internal AMBA AHB bus and give access to the complete set of registers. The SPI and General purpose I/O interfaces are accessible through the router's configuration port, which allows SPI devices to be accessed, and general purpose signaling to be performed directly through RMAP commands, or through the UART and JTAG interfaces. An auxiliary time- / interrupt-code interface is present, for sending and receiving time- / interrupt-codes through external pins. Parts of the interface use dedicated pins, while the rest are multiplexed on the general purpose I/O pins. For more information, see the advanced datasheet for the GR718 [5].

IV. Packet Routing Features

The router's switch matrix can connect any input port to any output port. Access to each output port is arbitrated using a round-robin arbitration scheme based on the address of the incoming packet. A single routing-table is used for the whole router, where access to the table is arbitrated using a round-robin scheme based on the input port number. Both addresses and input port can be assigned either high or low priority.

All the addressing modes, such as path, logical, and regional logical addressing are supported. Group adaptive routing is fully supported, meaning that both path and logical addresses can be individually configured to use one or more output ports. A unique feature is the support for packet

distribution, which can be used to implement multicast and broadcast addressing. Also packet distribution can be enabled for any address. Each router port is equipped with a timer which can be individually enabled/disabled. The timer can be used to recover from potential deadlock situations resulting from either a stalling source node or stalling destination node.

V. SpaceWire standard revision 1 support

Three changes were identified as having a technical impact on the GR718B development. The first one is the addition of timers in routers. The GRSPWROUTER IP core already contained programmable packet timers for each port, which meant that no changes were required. However, an addition to the functionality was made in order to be able to distinguish between overrun and underrun timeouts. The second change is a modification of the link interface FSM. Two requirements have been identified that potentially can cause the SpaceWire codec to make unwanted transitions. These are unlikely corner cases and very few if any problems have been seen in practice. This modification will probably not affect backward compatibility with older SpaceWire codecs, so the risk of including this modification in GR718B was estimated to be very low.

The final and most complicated change was the addition of distributed interrupts. The distributed interrupt scheme introduced two new control codes, called interrupt-code and interrupt-acknowledge-code, which uses one of the reserved control bit combinations of Time-Codes. It must therefore be made sure that they cannot interfere with the normal Time-Code facilities. All existing devices might not be forward-compatible with revision 1 compliant devices due to the interrupt- / interrupt-acknowledge-codes.

The distributed interrupt scheme was identified as part of revision 1 that caused the highest implementation risk if included in GR718B. Therefore, the router was made flexible enough to allow ports' handling of the new control codes to be configured individually. In this way the router can be used as a device that enables old and new equipment to be used in the same SpaceWire network.

The distributed interrupt scheme has been defined by [6], and GR718B supports all of the requirements put on routers, as well as some optional features to minimize the effects of errors such as a “babbling idiot”. Due to the uncertainty regarding some details in the specification, GR718B was given a high degree of configurability on how to handle the distribution of interrupt- / interrupt-acknowledge-codes.

VI. SpaceWire-D support

There is a new emerging protocol called SpaceWire-D, where D stands for deterministic [7]. This is anticipated to be widely used in the future to provide deterministic and low-latency transfer of control and command information while still preserving the high bandwidth of SpaceWire. It basically consists of a time-slotting table replicated in each unit (node or router) in the SpaceWire network. Therefore, a router needs to have support for SpaceWire-D if it is to be used in a network utilizing that protocol. GR718B implements support for SpaceWire-D by monitoring packet transfers. In the case of a packet being transferred while a Time-Code is received, the packet is truncated and an EEP is inserted at the end of the packet. The truncation can be individually enabled/disabled per port, and there is a programmable Time-Code filter per port as well. The filter allows for each port to have different Time-Code values or ranges that truncates packets. The programmable filters also allow distributed interrupt-codes to truncate packets.

GR718B implements status bits that inform software if a packet has been truncated due to a received Time-Code. There is also an option to automatically send an interrupt-code when the truncation occurs

VII. SpaceWire Plug-n-Play support

SpaceWire Plug-and-Play allows SpaceWire routers and nodes in a network to be identified and configured. This is defined by [8]. The standard uses RMAP commands and replies for communication, but with a different protocol ID.

GR718B includes basic support for SpaceWire Plug-and-Play, which covers device identification and support for network discovery. Extended capabilities, such as routing table configuration, and port configuration through SpaceWire Plug-and-Play, were not included due to the fact that the standard was not considered mature enough at the time of implementation. The SpaceWire Plug-and-Play functionality can be disabled by means of a configuration pin.

VIII. SpaceWire in-system test

A built-in self-test is provided for the verification of the SpaceWire router and codec functionality. The SpaceWire In-System Test (SIST) protocol provides the means for verifying larger part of the designs' functionality without the need to generate high speed test patterns and observe results at high frequencies.

The internal SIST module is connected to the router via a dedicated FIFO port. The external side of the SIST module is connected to the AMBA APB bus, which is only accessible through the JTAG and UART (debug-) interfaces. Thus it is not possible to configure the SIST module via a SpaceWire link.

The SIST module can generate and send SpaceWire packets via the internal FIFO port. It can also receive SpaceWire packets via the FIFO port and check their contents. The packets are generated deterministically and can therefore also be easily checked on reception.

IX. Power-saving features

The GR718B incorporates the following power saving functions:

- Disabling of unused on-chip LVDS receivers/transmitter
- Disabling of unused off-chip LVDS receivers/transmitter or repeater devices
- Clock-gating of unused SpaceWire ports

The existing power-down functionality provided for the LVDS I/O cells in the DARE+ library are utilized for this purpose. Signals for disabling the off-chip LVDS devices are shared with the external pins provided for general purpose I/O. It is, therefore, possible to control up to 18 external LVDS devices, with one external pin per device.

X. Technical overview

The GR718B has been designed for operation over the full military temperature range -55°C to +125°C and it is powered with a nominal supply of 1.8V (Core) and 3.3V (I/O). The GR718B is available in a 256 pin CQFP package. The system clock domain can operate at a maximum frequency of 50MHz, while the SpaceWire clock domain can operate up to 200MHz. This is possible as the GR718B includes a radiation hardened-by-design PLL. The PLL allows the use of either an external input clock to the SpaceWire clock domain or the system clock itself. Further details are provided in the product datasheet and user manual [5]

XI. Qualification and testing

The GR718B has undergone an extensive validation process which included RTL simulations, electrical and functional testing at high operational frequencies. The production test program developed for this component includes over 5000 different electrical tests to guarantee that each flight device is compliant with the information provided in the datasheet.

The GR718B is currently being qualified for space applications following a lot validation approach. The first flight units are expected to be available in January 2017. Prototype devices are already available now. The screening and qualification flows selected for the flight units are based on test methods 5004 and 5005 of the MIL-STD-883K standard for class level S components. These flows cover all of the tests required by ESA's generic specification ESCC9000. With the use of the GR718B's in-built SIST protocol, flight devices will undergo dynamic burn-in as part of the screening flow, i.e. the devices will be fully exercised during testing. The same approach will be utilized during high temperature operational life test, which will be done as part of the qualification program.

A prototype board has been developed together with Pender Electronic Design (Switzerland). The board, shown in Figure 3, comprises a custom designed PCB in a 6U Compact PCI format, making the board suitable for stand-alone bench top development, or if required, to be mounted in a 6U CPCI Rack. The purpose of this board is to provide developers with a convenient hardware platform for the evaluation and development of software for the GR718. The principle interfaces and functions are accessible on the front and back edges of the board. Secondary interfaces are accessible via headers on the board.

XII. Radiation tolerance

The technology used for the development of the GR718B has been hardened-by-design against the impact of ionising radiation. The designed Total Ionising Dose tolerance for the GR718B is 300 krad(Si). This tolerance is suitable for most space missions. Results recently obtained on a test vehicle built with the same library support this design tolerance. Additional Total Ionising Dose testing of the GR718B is currently scheduled as part of the qualification flow.

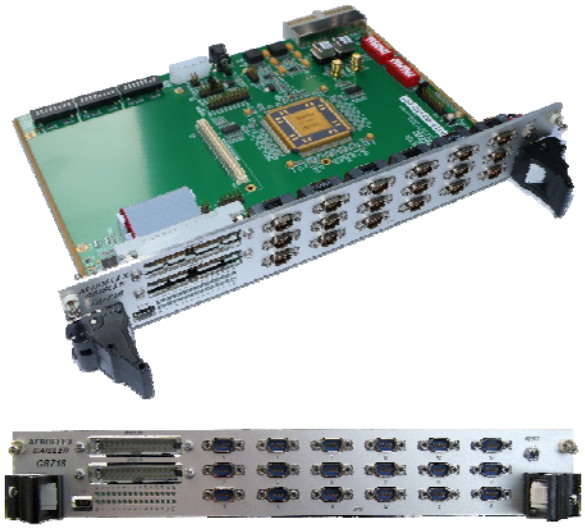


Fig. 3. *GR718B prototype board*

The technology used for the development of the GR718B has been proven to have a high Single Event Latch-up tolerance, with a minimum LET value of 118MeV.cm2/mg. Furthermore, the SpaceWire router has been almost entirely implemented using Single Event Upset hardened-by-design flip-flops. Only the signal routing functions of the SpaceWire port receivers have been implemented with much faster, unhardened, FFs. However, a parity protection scheme has been implemented on these unhardened FFs to deal with any undesired effects from the radiation environment.

Static and dynamic Single Event Effect testing of the GR718B SpaceWire router has been performed using one of the prototype boards as shown in figure 4. This board allowed to: remotely configure the device under test; link all SpW ports in a single Daisy chain (to fully exercise the device during testing); to access the SIST port to send data via the SpW ports (dynamic testing) and to monitor for any possible radiation induced errors.

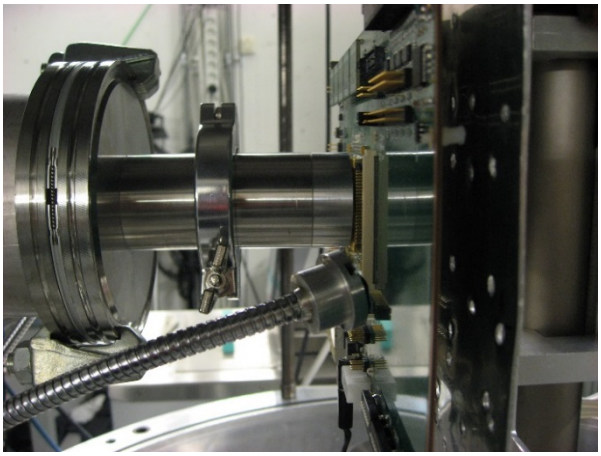


Fig. 4. *GR718B Single Event Effects testing*

XIII. SPI Controller and interface

The SPI controller provides a link from the router configuration port to an external Serial Peripheral Interface (SPI) bus. The SPI bus parameters are highly configurable via registers. The SPI controller features configurable word length, bit ordering, clock gap insertion, and automatic slave select.

The external SPI bus can be used for connecting to external components using the SPI interface. Typical user scenario can be to remotely monitor the temperature close to where the GR718B is mounted.

XIV. General purpose interface

The GR718B is equipped with 24 general purpose inputs and outputs accessible from the router configuration port. The general purpose interface can be programmed to have one of the following functions:

- Individual general purpose inputs or outputs
- Distribution of time-codes to companion devices
- Extended state or error signaling to companion devices
- SpaceWire transceiver enable per port for off-chip transceiver support
- Extend number of SPI slave select signals supported

XV. Debug and integration

GRMON2 [9] is a debug monitor used for develop and debug GRLIB systems. The debug monitor GRMON2 has extended support for the GR718B and has built-in commands for configuring and debugging the SpaceWire router core. The GR718B configuration and its peripherals are accessed on the AMBA bus through a JTAG or UART debug-link.

Connection via SpaceWire RMAP interface is supported as long as the SpaceWire has RMAP and automatic link start. An Ethernet to SpaceWire Bridge [10] is required to tunnel SpaceWire packets from the Ethernet over to SpaceWire.

XVI. Results

Although the targeted speed in the design of the SpaceWire links was 200 Mbit/s, during functional testing and validation of the prototype devices, it has been found that these ports are able to operate successfully well above 200 Mbit/s. The typical power consumption has been found to be below 3W, when running all of the 18 SpaceWire ports at 200 Mbit/s.

Assembly of the first GR718B flight lot has now been successfully completed and the screening as well as qualification tests are under way. The first flight units are expected to be available in January 2017. Electrically verified prototypes can be ordered directly from Cobham Gaisler [1].

Single Event Testing of the SpaceWire router has demonstrated its suitability for Space. The data collected has confirmed the Single Event Latch-up insensitivity of the device and the suitability of the radiation hardened PLL. The extensive data collected has also shown that the device is compliant with the SPW standard of having a bit error rate (BER) below 1E-12, when operated under a worst case configuration (i.e. minimum supply and maximum operational frequency) in a GEO environment. Furthermore, the majority of Single Event Upsets induced errors detected during testing may be handled either by the SPW protocol or by implementing a CRC scheme.

XVII. Conclusion

The high number of ports, together with the wide range of supported functions and the high configurability of the GR718 should make the device suitable for most current and future SpaceWire networks. The first lot of space qualified GR718B are expected to be available in January 2017.

During the GR718B development, Cobham Gaisler has participated and contributed to the ongoing standardization work of the distributed interrupt scheme that will be part of the SpaceWire standard revision 1, as well as the upcoming SpaceWire Plug-and-Play standard. These extra efforts are expected to pay off with an advanced multi-port SpaceWire router ASIC which enables coexisting of older and newer equipment in the same network.

The currently available radiation results support the selection of hardened-by-design features implemented in the SpaceWire router and confirm its suitability for space applications.

XVIII. Acknowledgement

We would like to thank the support of European Space Agency in the development and radiation testing of the GR718B SpaceWire router. More specifically, we would like to thank: M. Suess and R. Dittrich, and for their technical support.

REFERENCES

[1] GR718-BOARD – GR718 Development Board User's Manual, 2016 User's Manual, April 2016

[2] ECSS - Space Engineering, SpaceWire - Links, nodes, routers and networks, ECSS-E-ST-50-12C, July 2008

[3] GR740 – Quad Core LEON4 SPARC V8 Processor, 2016 Preliminary Data Sheet and User's Manual, June 2016

[4] ECSS - Space Engineering, SpaceWire - Remote memory access protocol, ECSS-E- ST-50-52C, February 2010

[5] GR718B – Radiation-Tolerant 18x SpaceWire Router, 2016 Preliminary Data Sheet and User's Manual, April 2016

[6] Yuriy Sheynin, Distributed Interrupts in SpaceWire Interconnections, International SpaceWire Conference, Nara, November 2008 (outdated)

[7] SpaceWire-D - Deterministic Control and Data Delivery over SpaceWire Networks, Draft B, April 2010, ESA Contract Number 220774-07-NL/LvH

[8] ECSS - Space Engineering, SpaceWire Plug-and-Play protocol, ECSS-E-ST-50-54C Draft, March 2013.

[9] GRMON2 – Debug Monitor for LEON-based computer systems and SOC design based upon GRLIB IP library, 2016 User's Manual, June 2016.

[10] GRESB – SpaceWire/Ethernet Bridge with routing capabilities, 2016 User's Manual, April 2016.

SpaceWire Components, Long Paper

An IP Core for the SpW family of protocols

Antonis Tavoularis, Vassilis Vlagkoulis & Fotis Kostopoulos
TELETEL S.A., Athens, Greece
{A.Tavoularis, V.Vlagkoulis, F.Kostopoulos}@teletel.eu

Brice Dellandrea
Thales Alenia Space, competence Center Platform Integration
Brice.Dellandrea@thalesaleniaspace.com

Tam Le Ngoc
Airbus Defence and Space, BU Electronics, Elancourt, France
Tam.Lengoc@airbus.com

Luca Fossati, Jorgen Ilstad, David Jameux
European Space Agency,
ESTEC, Noordwijk, Netherlands
{Luca.Fossati, Jorgen.Ilstad, David.Jameux}@esa.int

***Abstract*—This paper presents the architecture, functionality, and performance of the SpW Interface Node IP which constitutes a configurable IP Core from which non-SpW experts can tailor a customized SpW interface for integration into their developments. The IP core offers numerous configurability options and is also expandable to include additional blocks as the SpW protocol family evolves.**

***Index: SpaceWire*, RMAP, SpW-D, NDCP, CPTP, TDP, SpW Router, Distributed Interrupts, AMBA.**

I. Introduction

TELETEL S.A., together with Thales Alenia Space and Airbus Defence and Space, have been working on the development of an IP Core under ESA contract 4000113046 (SpaceWire Node Interface IP Core) aiming at providing a solution helping non-SpW experts to integrate advanced SpaceWire functionality in their flight equipment. The IP Core contains RMAP ([2]), NDCP [5], CPTP ([3], [4]) and TDP ([7]) protocol engines and can be provided as a single SpW Port Node or a Node with an integrated SpW Switch. Each protocol block can be independently configured allowing the user to customize the implementation to its own requirements (size of memory blocks, maximum number of RMAP pending transactions etc.). In addition, the IP Core allows the user to include its own technology primitives for implementation on different target ASIC technologies.

The design has been validated on TELETEL's SpW Xilinx-based SpW G2 board and also on SkyLab's PicoSky development board for ProASIC3 meeting all the functional and performance requirements.

The rest of the paper focuses on RMAP/NDCP and CPTP and does not present design issues related to the SpW Switch design which was a readily available block or the TDP which is a third-party IP-Core integrated in the Node IP ([6]). Nevertheless, the results presented herein, related to timing and resources utilization, include all blocks.

II. Design Solution

The IP Core is based on the AMBA AHB bus specification ([9]), allowing seamless core integration to AMBA based systems (e.g. LEON SoCs). All protocol engines are connected to TELETEL's AHB DMA engine which offers user configurable number of DMA clients and minimum burst size of 8 cycles. TELETEL's RMAP core (developed in ESA contract 4000105444/12/NL/CBI) is being used, providing a response time in the range of one microsecond. The RMAP core has been extended to support NDCP, in order to reuse most of the logic resources and minimize utilization.

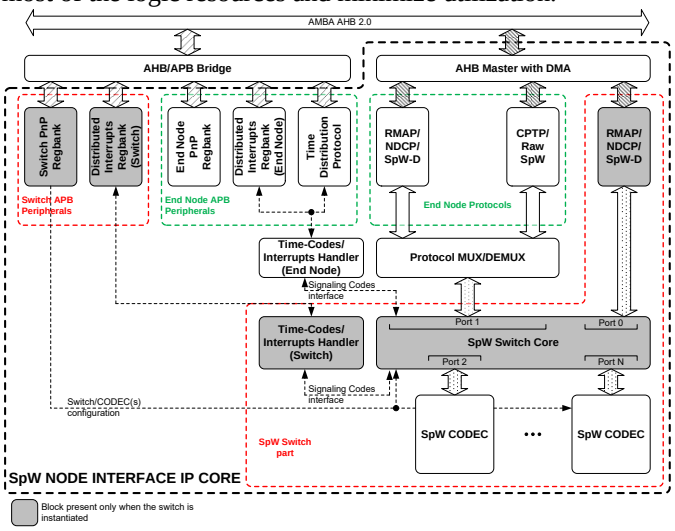


Fig. 1: Node IP overall architecture

For CPTP, a newly developed block has been integrated which offloads the user from time-consuming operations such as CRC/PEC calculation/verification and packet length verification.

The TDP block developed by COBHAM Gaisler [6] is integrated in the IP for SpaceWire Time-Distribution.

Depending on the user needs, the IP Core can instantiate a SpW Switch core with user configurable number of ports and RMAP/NDCP support for SpW Switch configuration.

Finally, the core can be delivered with TELETEL's SpaceWire CODEC and also provides the de-facto standard 9-bits FIFO SpaceWire CODEC interface allowing the user to instantiate its own or third-party SpaceWire CODEC.

A. Architecture

The Node IP has different configurations spanning from SpW node with a single link to multiple links with a SpW Switch instantiated, having or not having support for NDCP, TDP etc. The top level architecture of the Node IP core consists of the following blocks (shown in *Fig. 1*):

- One or more SpW CODECs
- A non-blocking SpW Switch which switches packets and signaling codes among the external SpW interfaces and between the external interfaces and internal ports.
- A protocol MUX/DEMUX block which discriminates received packets, dispatches them to the appropriate protocol engine (receive direction) and multiplexes them (transmit direction).
- A RMAP block instantiated as Initiator, Target or both, used to handle RMAP transactions and execute RMAP commands. The RMAP Target is available from a previous study and has been designed with performances meeting the strict timing requirements for SpW-D. Within the study the core was extended to handle NDCP commands. Two such blocks are currently instantiated; one for the Switch and one for the End Node, which can have different parameters (verify buffer size, RMW logic, authorization keys etc.).
- The CPTP/Raw SpW block which handles CPTP and Raw SpW packets (packets not handled by any instantiated protocol engine).
- The Time Distribution block which handles SpW-TDP protocol which is implemented as an APB peripheral.
- The SpW Time Codes/Interrupts Handler block(s) which handles the Signaling Codes at both the End Node and the Switch.
- A single AHB Master which is responsible for DMA read/write operations to/from the system memory. The engine supports ATOMIC transactions required for RMAP RMW and NDCP CAS. A single AHB Master is implemented, which has a configurable number of channels and handles the requests for both the End Node and the Switch.
- APB peripherals which host the NDCP, CPTP, RMAP, Switch configuration registers, SpW Interrupts configuration & status registers as well as the NDCP address translation block.

B. User configuration options

The core supports various options, configured either through VHDL generics or IP-XACT, e.g. number of SpW ports, data bus width, target technology etc. and also supports configuration options separately for each block such as the implementation of NDCP translation and RAM or ROM, maximum SpW packet size, RMAP memory map etc. The only technology dependent primitives required are for the SpW LVDS transceivers and the asynchronous FIFOs, both of which constitute part of the SpW CODEC. A non-exhaustive list of the core's configuration options follows:

- Overall Core: target technology, number of SpW ports, NDCP support, TDP support, SpW Switch instantiation, Switch NDCP support, data bus width.
- RMAP: Initiator/Target support, Initiator/Target DMA length, Verify buffer size, Target supported commands and parameters all independent for the End Node and the Switch, maximum number of Initiator pending transactions (End Node only).
- NDCP: Vendor/Device IDs, versions, RAM/ROM based address translation, address translation table depth, base transmit rate/range/divider etc. independent for the End Node and the Switch.
- CPTP: Tx/Rx descriptors width/depth, maximum packet/DMA size, discard Rx packet on memory buffer unavailable (End Node only).

C. CPTP Block architecture

CPTP operation supports transmission/reception of multiple packets without occupying the user (e.g. host processor), other than for initializing the memory structures as described below.

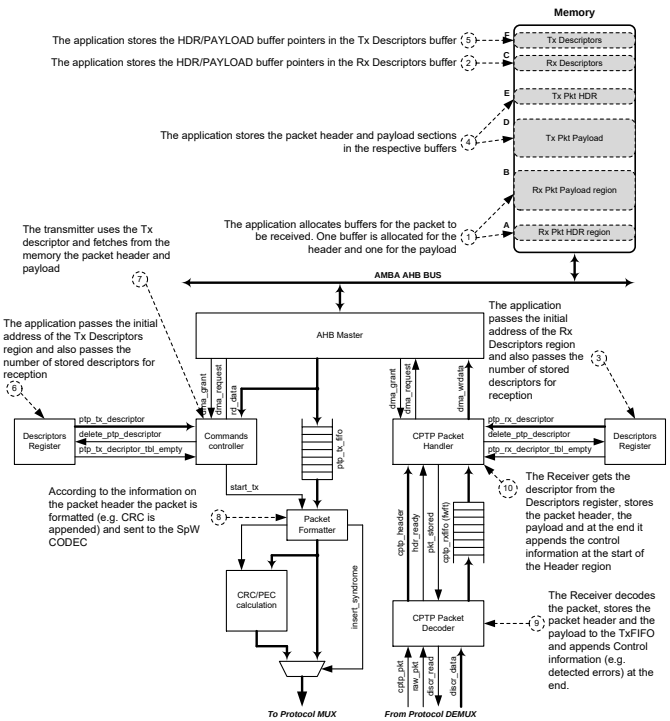


Fig. 2: CPTP block architecture & operation

Packet headers and payloads can be stored at different memory locations and separate areas exist for storing the Tx and Rx descriptors. In the transmit direction, the user stores the packets for transmission in the memory and it then writes to the Tx Descriptors memory area the pointers to the Tx packets. Finally the user writes to the CPTP block the number of Tx pointers and from the packets are automatically transmitted by the block without any user occupation. In the receive direction

the approach is similar, with the user allocating the required memory areas and pointers and downloading the number of available buffers to the CPTP block as shown in *Fig. 2*.

Update of the descriptors can be done during initialization, or even after transmission has started. In the latter case, after storing the header and payload segments of the packets in memory the application updates the descriptors area and the passes to the CPTP block the number of additional descriptors. Update of the location of the Descriptors area is not required as this is a circular buffer in the memory and the block simply fetches the next transmission descriptor when the user informs the block that additional N descriptors have been added.

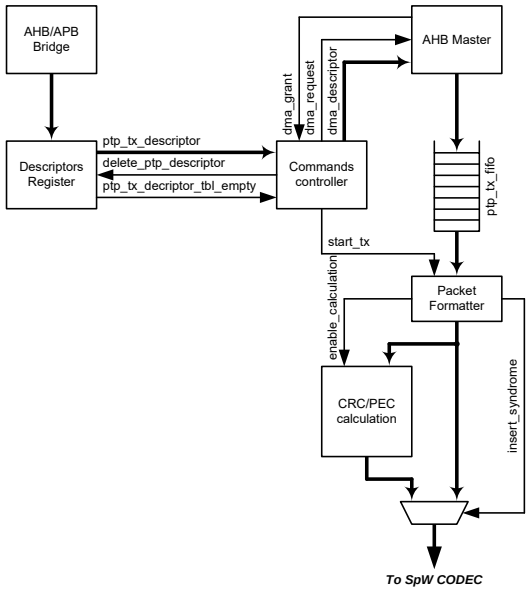


Fig. 3: CPTP Block Transmitter

The architecture of the transmitter is shown in *Fig. 3*. It consists of the following modules:

- The Descriptors Register: Block which contains the current address to be accessed in the Descriptors Area in order to fetch the header/payload pointers for the next packet. The module contains the number of currently available transmission descriptors. It is programmed by the user through the AHB/APB bridge.
- The Commands controller: Module which fetches the packet header and payload from the memory for transmission. It stores the fetched data in the transmission FIFO (cptp_txfifo).
- The Packet formatter: Module which reads header and payload from the transmission FIFO, converts the FIFO words to 9-bit SpW CODEC interface words, enables the CRC/PEC calculation block when required and appends the calculated CRC/PEC to the CPTP packet. If the packet under transmission is a Raw SpW packet (information contained in the packet header) no CRC/PEC is appended.
- The CRC/PEC calculation block: Module fed by the Packet Formatter with data to be transmitted and calculates the CRC/PEC to be appended to the end of the SpW packet.

The Receiver handles both CPTP and Raw SpW packets. It consists of the following blocks (shown in *Fig. 4*):

- The Descriptors Register: Block which holds the pointers for the packet header and payload. It is the same block as the one used for the CPTP transmitter
- CPTP Packet Decoder: Block which captures the packet header, stores the payload in a FIFO (CPTP Rx FIFO), truncates packets which are longer than the programmed size, identifies packet errors (e.g. CPTP length, CRC/PEC error), discards or stalls reception (programmable function) if no Rx descriptors exist.
- CPTP Packets Handler: Block which is responsible for requesting ownership of the system bus (through the AHB Master) in order to store the received packets in the system memory. The Handler, acquires the descriptors from the Descriptors Register, stores the payload in the memory, stores the header and finally, informs the CPTP Packet Decoder that the packet has been stored in the memory and a subsequent packet can be received.

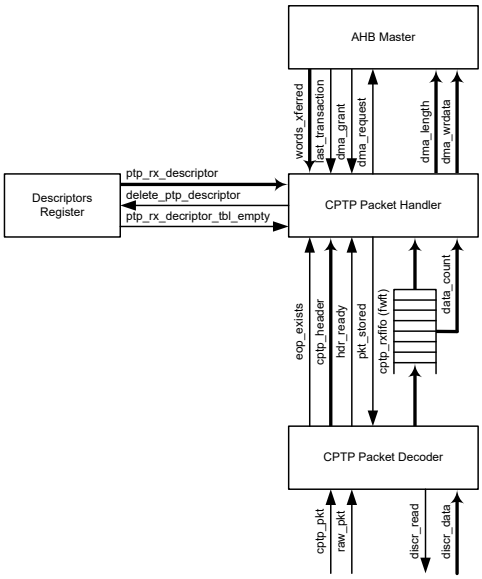


Fig. 4: CPTP Block Receiver

The block supports user notification upon programmable events as for example when a programmable number of packets have been received or upon time-out.

D. NDCP Address Translation

The architecture and operation of TELETEL's RMAP Core has been previously presented and herein only the extensions required for NDCP support are presented.

The NDCP specification specifies a protocol based on RMAP syntax (with different semantics) and a Network Management Service which enables a peripheral device to expose its capabilities in order to allow for discovery and configuration by a control device. The latter, includes a set of resources which allow controlling a peripheral device in a

standardized way in order to permit for interoperability between devices.

If directly mapped to memory addresses, NDCP field identifiers would result in a highly fragmented memory space. In order to support a contiguous memory space, the NDCP block contains an "Address Translation" block which receives the Application Index, Protocol Index, FieldSet ID and Field ID from the received NDCP packet and returns the address to be accessed in the memory space. The Address Translation block performs translation between the fragmented memory space created by the NDCP field identifiers to a more flat memory space in order to ease implementations and offer better memory usage by the Node IP.

The Address Translation block consists of three entities:

- The Translation Table, which contains the relationships from the NDCP field identifiers to the Node IP memory space. It can be implemented either in RAM or ROM with the RAM version being able to be programmed through the AMBA from a host processor and the ROM version being targeted for simple End Nodes or Switches.
- The Translation Logic block which receives NDCP packet header fields and searches in the table for a matching entry. Upon match, the translated address is returned to the RMAP Target logic in order to write/read the addressed field(s).
- The Device Ownership logic which assesses whether the received packet was sent by the peripheral owner or not for write transactions.

The Address Translation Table contains entries which are used to map the fields of an incoming NDCP packet to the physical address which shall be accessed. Specifically, each line contains the following fields:

- The address parameters: Contains the address parameters of a set of fields which can be an entire FieldSet or a subset of a FieldSet. It contains the following fields:
  - Application Index: Application Index for this entry
  - Protocol Index Protocol Index for this entry
  - FieldSet ID: FieldSet ID for this entry
  - Field ID: Field ID number of the "lowest" Field Identifier for this entry. It is used to identify whether the requested access is performed outside the FieldSet's boundaries (e.g. if the NDCP packet contains a Field ID of value which is less than the table entry, the access is considered as outside the boundaries).
- The Access Parameters:
  - FieldSet length: Length of the FieldSet (or length of the FieldSet subset). It is used to identify whether the requested access is performed outside the FieldSet' s boundaries (e.g. if the NDCP packet contains a Field ID of value which is more than the table Field ID value and FieldSet length added together, the access is considered as outside the FieldSet boundaries).
  - CAS FL/RO FL: CAS/RO FL is set to 1 if this region is followed by a CAS modifiable/Read-only region. The CAS FL and RO FL fields are applicable to writable regions only and required for the following reason: assuming that a Write operation starts in the region and spans more than its upper boundary then this shall be allowed if the region above is non-existent (write data is discarded) but shall generate an error (Read-Only Field or Field Not Writable) in case the region that follows is Read Only or CAS modifiable
  - RO: Used in case of successful match. It indicates that the respective FieldSet is Read-Only.
  - RSV: Used in case of successful match. It indicates that the respective FieldSet is reserved. This is used 1) in the case of the Vendor String FieldSet (to allow for the FieldSet to be left unimplemented) and 2) for the reserved region of any other FieldSet at the FieldSet end. It is used by the RMAP/NDCP protocol engine so that all zeros are returned in the Reply packet.
- The Physical Address: Contains the Physical Address of the entry. This is the address of the FieldSet/Field within the node's memory map and will be returned to the RMAP/NDCP block in order to perform the access.

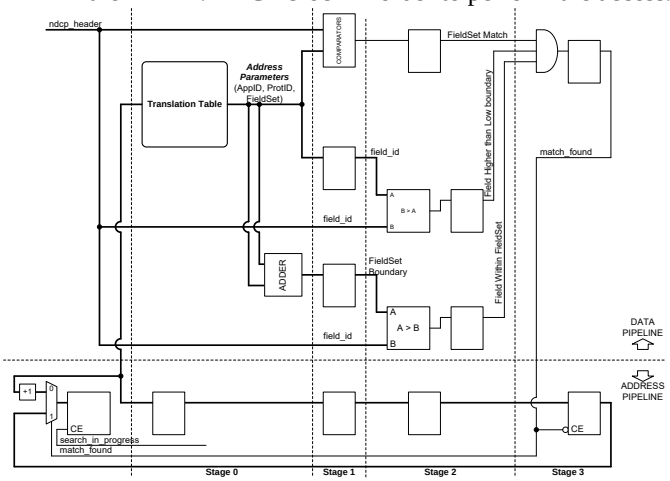


Fig. 5: NDCP Translation Logic architecture

The Address Translation Logic block operates in two stages. Upon the reception of a NDCP packet it first searches the table for a matching entry and, upon a successful match, it checks the access rights in order to determine whether the requested operation shall be performed or not (e.g. a write is requested to a Read-Only FieldSet). Since it is the block that will either authorize or not the requested operation, it also checks the protocol fields such as Target Logical Address, Key, Data Length etc. It scans the table and performs comparison of the NDCP packet fields with each table entry fields for matching entry. When a matching entry is found, the address of this entry is latched in the Translation Table Read

Address vector so that the access rights and the Physical address of the matching entry appear on the table output.

Comparisons are performed on large vectors which would introduce combinational delays and reduce the Core's operation frequency if they were done in a single clock cycle. Alternatively, searching and matching the access rights for each entry may require several clock cycles and for a large number of entries this will reduce the overall performance. To this respect, the search logic is implemented in pipelined fashion as shown in *Fig. 5* and in the timing diagram of *Fig. 6*

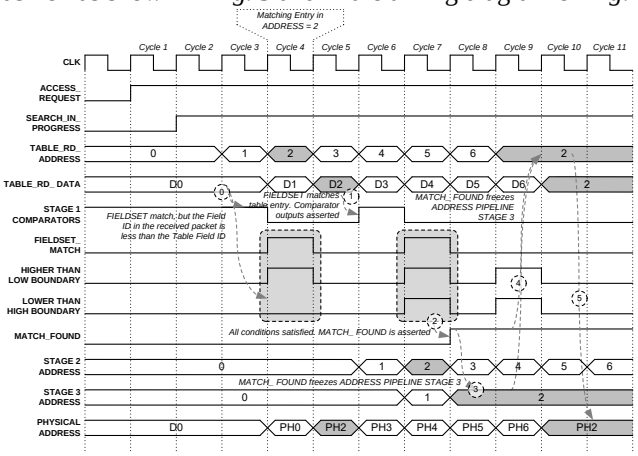


Fig. 6: NDCP Translation Logic operation

The following paragraphs explain the way in which the Translation Table shall be programmed for proper NDCP block operation.

The example in *Fig. 7* shows a FieldSet which spans from Field O to Field U and the rest (Field U+1 up to Field V) is non-exiting fields (reserved fields).

The table is scanned sequentially and when the first matching entry is found the respective physical address and grant/deny signal is returned to the RMAP/NDCP block. In addition, when FieldSet is matched, the logic checks whether the Field_ID of the received command is within the boundaries of the region found in the table and if they are found to be outside the command is not granted.

The order in which entries are inserted in the translation table is important for the correct operation of the logic. In the example shown in *Fig. 7* if the RO region was the first entry in the translation table then all write to regions 1 to 4 would be denied. To this respect the user shall insert all writable regions first and RO regions at the end of the table.

In addition, the NDCP specification ([5]) mandates that a write to a reserved region shall be accepted but no write shall be performed. If we program only regions 1 to 3 in the table, a write to the reserved space (command G in the example) will not produce a match and a reply status FieldSet reserved will be returned in the NDCP reply. To this respect for each FieldSet the table shall have as its first entry the reserved region which lies at the end of the FieldSet which shall also be marked as reserved (RSV field shown in *Fig. 7*).

Region 4 is also programmed as read-only region for the following reason. If a write access is performed to this reserved region then the access will be authorized since a match will be found in the first table entry. If a read however is performed then the command will not be matched with the first entry since the RO field is not set. Consequently there must be an entry which defined that the region is also readable which is the fifth entry in the table.

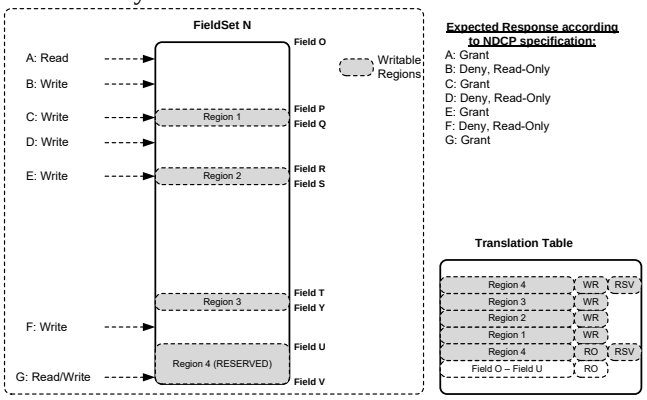


Fig. 7: NDCP Translation Table example

Finally, each FieldSet is readable to its entirety (actual size) by all devices. This means that the table shall have an entry which corresponds to the actual FieldSet extents. This region is marked in the table as Read-Only section and is programmed as the last one in the table (Field O to Field U). This region is marked as RO. The reason that the last two entries are not merged to one entry (entire FieldSet) is that in case a read crosses the boundary of the reserved region, the block shall have a way to inform that zeros shall be returned as mandated in the NDCP specification for reserved regions.

E. SoC Integration & Reliability

The design of the core has taken into account issues like its integration in SoCs as well as reliability issues. As such it supports different interrupts to the host processor, time-outs on receive and transmit direction, packet truncation, selectable consumption of received packets in case receive descriptors are not available etc. A non-exhaustive list of these features is:

- Interrupts on illegal conditions: NDCP. RMAP illegal field identifier, NDCP non-owner attempt, write attempt to CAS, RO NDCP fields, RMAP Reply TID not found, RMAP Reply timeout, RMAP Reply rejected, RMAP error received (Header CRC, illegal command, Early EoP etc.), CPTP CRC/PEC or Length error, Tx/Rx time-out etc.
- Interrupts on nominal conditions: CPTP/SpW packet group transmitted/received, RMAP/NDCP command executed, RMAP Reply received, expected SpW Interrupt(s)/Acknowledgement(s) received.
- Reliability options: discard CPTP/SpW packet if no pointers exist, Flush packet and append EEP if max length is exceeded or Rx/Tx timeout occurs, support for statistics (packet/NCHARs received and their rate).

F. Core extensibility

The core has been designed with the requirement to be easily extensible in order to support additional protocols as the family of SpW protocols evolves. In order to add a new protocol engine the user shall add its implementation between

the protocol MUX/DEMUX and the AHB DMA engine. All these blocks support programmable number of channels, through VHDL generics or IP-XACT, and the DEMUX block also supports mapping of PID to the newly added channels for dispatching of received packets to the added protocol engines.

III. VALIDATION APPROACH

The IP core was implemented on two different technologies, Virtex 6 and ProASIC3. The boards in which the IP core instances were deployed were respectively TELETEL's SpW G2 board and SkyLabs PicoSky board.

Three IP core instances were implemented in the Virtex 6 target: one instance with a six-port SpW Switch (four ports external to the SpW Node IP core) and two minimal implementations with no SpW Switch. An external DDR2 memory was attached to the AHB bus, through a memory controller, to support read/write access through RMAP and NDCP (e.g. Vendor String). The system clock frequency was 125 MHz.

In the ProASIC3 target a single instance of the IP was implemented, with 4-Ports SpW Switch instantiated, since the board offers only two SpW connectors. An external SRAM memory was attached to the AHB bus and the system clock frequency was 20 MHz. Both implementations were tested at SpW link speeds up to 200 Mbps.

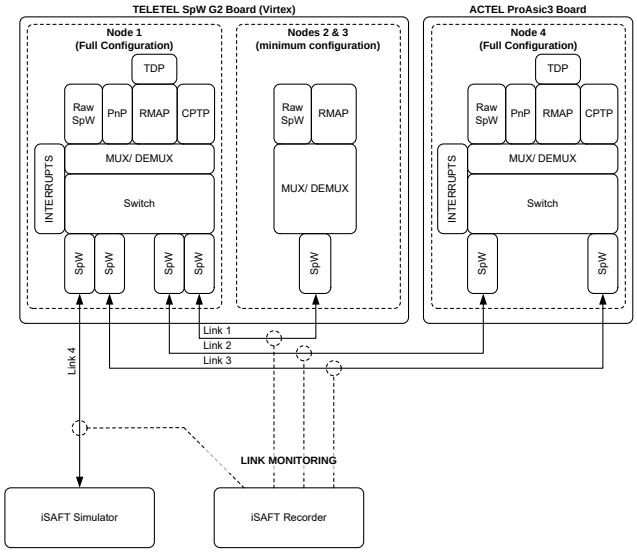


Fig. 8: Topology used for validation/demonstration

Validation was based on the iSAFT tool chain which is an integrated solution for validation of on-board protocols and devices. For the purposes of the project, the SpaceWire simulator part of iSAFT is being used which supports transmission/reception of SpaceWire packets to the connected Unit Under Test (UUT) and assessment of the UUT's response. At the same time, it allows the user to examine all exchanged transactions by decoding them through the WireShark protocol analyser. The tests covered both functional and performance validation as well as injection of errors for all protocols to assess design's robustness.

During all tests the traffic was also captured by an iSAFT Recorder whose accurate time-stamping protocol decoding capabilities helped towards both the functional and performance evaluation of the IP under test. The scenarios included stress tests with continuous back-to-back traffic and duration up to 22 hours and validation was performed through sample windows to validate the correct response is returned from the UUT and through the Recorder's statistics capability (total packets/bytes transmitted/received) to validate that no information was lost.

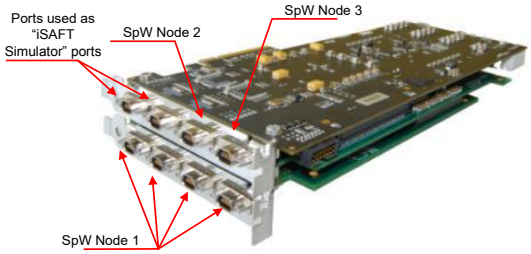


Fig. 9: TELETEL's SpW G2 board used for Node IP validation

Functional validation was performed incrementally. Tests started with the validation of the NDCP implementation on a Node IP instance with no SpW Switch through a point to point connection between the iSAFT Simulator and the Node IP instance. The IP core instance was configured through NDCP commands after the iSAFT simulator got ownership of the device, and the configured values were read back through RMAP. All NDCP fields were read and all writable fields were configured. Error injection at NDCP level and repetition of the nominal tests right after validated the robustness of the core.

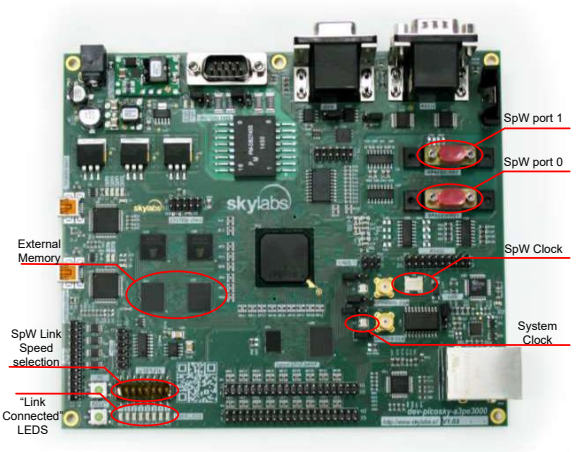


Fig. 10: SkyLab's PicoSky board used for Node IP validation

Validation of the SpW Switch's NDCP block was performed in a similar way. Tests included configuration of the switching matrix, GAR, self-addressing, time-outs etc. Stress tests with the traffic generation capability of the iSAFT Simulator followed. A series of mixed RMAP/NDCP commands were transmitted back-to-back to both the End Node and the SpW Switch in order to validate the robustness under heavy load and assess the performance.

Validation of the CPTP block was also performed through RMAP. Download/upload of packet header and payloads, Tx/Rx descriptors and access to the descriptors registers was performed through RMAP. The tests included transmission of

SpW and CPTP packets from the iSAFT Simulator to a Node IP instance with Switch, transmissions in the opposite direction and transmission/reception between IP instances. Once again, the traffic generation capability of the iSAFT simulator was exploited in order to periodically stimulate the IPs and validate the block under the presence of continuous traffic.

IV. PERFORMANCE RESULTS

A. Latency

Performance tests were performed on IPs with and without the SpW Switch instantiated on both Virtex and ProASIC3 targets (results shown are with 100 Mbps link speed) and included the latency measurement from:

- The reception of a RMAP command, transmitted from the iSAFT Simulator, to the time the response was transmitted by the UUT (RMAP Target Latency)
- The stimulation (through a RMAP command) of the RMAP Initiator, to the time the RMAP command transmitted was transmitted (RMAP Initiator latency, currently measured on Virtex target only)
- The reception of a NDCP command, transmitted from the iSAFT Simulator, to the time the response was transmitted by the UUT (NDCP Latency)
- The stimulation (through a RMAP command) of the CPTP Transmitter, to the time the CPTP/Raw SpW Packet was transmitted by the UUT (CPTP latency)

TABLE I. MEASURED LATENCIES ON VIRTEX 6 TARGET

	With Switch (us)	**Without Switch (us)**
RMAP Target	1.58 us (SDRAM read) 1.24 us (reg. Write, RMW)	1.30 us (SDRAM read) 1.0 us (reg. Write, RMW)
RMAP Initiator	2 us (incl. SDRAM access)	1.76 us (incl. SDRAM access)
NDCP target	1.35 us (Read/Write) 1.4 us (CAS)	1.10 us (Read/Write) 1.15 us (CAS)
CPTP	3.25 us (incl. SDRAM access)	3 us (incl. SDRAM access)

TABLE II. MEASURED LATENCIES ON PROASIC3 TARGET

	With Switch (us)	**Without Switch (us)**
RMAP Target	6.4 us (SRAM Read/Write)	4.96 us (SRAM Read/Write)
NDCP target	6.6 us (reg. Read), 7.3 us (reg. Write, CAS)	5.1 us (reg. Read), 5.8 us (reg. Write, CAS)
CPTP	12.4 us (incl. SRAM access)	10.9 us (incl. SRAM access)

B. Synthesis results

The tables below present the synthesis results for both Virtex 5 and ProASIC3 targets synthesized with Synplify 2015.9. The tables that follow present the results for RMAP with NDCP support, for the CPTP block and for the overall IP. In addition, the results for the NDCP address translation are presented separately for various translation table depths for the End Node (RAM implementation) and for various numbers of ports, for the SpW Switch (ROM implementation).

TABLE III. RMAP WITH NDCP SUPPORT, VIRTEX 5 RESULTS

	RMAP with NDCP support, Verify buffer size 4K, Max DMA length 128 Bytes, Virtex 5 FX target		
	LUTs	***Registers***	***BRAMs***
RMAP Block	2072	1775	3
RMAP Initiator	941	831	1
RMAP Target	767	647	1
RMAP Packet Validator	342	242	1
Statistics Block	22	55	-

TABLE IV. RMAP WITH NDCP SUPPORT, PROASIC3 RESULTS

	RMAP with NDCP support, Verify buffer size 4K, Max DMA length 128 Bytes, ProASIC3 target		
	Core Cells	***Registers***	***BRAMs***
RMAP Block	8564	1851	21
RMAP Initiator	3842	703	11
RMAP Target	3614	738	2
RMAP Packet Validator	1415	351	8
Statistics Block	127	60	-

TABLE V. NDCP ADDRESS TRANSLATION, END NODE, VIRTEX 5 FX RESULTS

	NDCP Address Translation, Virtex 5 FX target		
Translation table Depth	***LUTs***	***Registers***	***BRAMs***
64	348	209	5
128	347	211	5
256	349	213	5

TABLE VI. NDCP ADDRESS TRANSLATION, SPW SWITCH (ROM IMPLEMENTATION), VIRTEX 5 FX RESULTS

	NDCP Address Translation, Virtex 5 FX target		
Number of SpW Ports	***LUTs***	***Registers***	***BRAMs***
4	344	209	5
8	344	209	5
16	344	209	5
32	344	209	5

TABLE VII. NDCP ADDRESS TRANSLATION, END NODE, PROASIC3 RESULTS

	NDCP Address Translation, ProASIC3 target		
Translation table Depth	***Core Cells***	***Registers***	***BRAMs***
64	1452	245	-
128	1472	249	-
256	1575	267	-

TABLE VIII. NDCP ADDRESS TRANSLATION, SPW SWITCH (ROM IMPLEMENTATION), PROASIC3 RESULTS

	NDCP Address Translation, ProASIC3 target		
Number of SpW Ports	***Core Cells***	***Registers***	***BRAMs***
4	1506	257	-
8	1506	257	-
16	1506	257	-
32	1506	257	-

TABLE IX. CPTP, VIRTEX 5 RESULTS

	CPTP, Max packet size 64K, Max Tx/Rx descriptors 16, Max DMA length 128 Bytes, Virtex 5 FX target		
	LUTs	***Registers***	***BRAMs***
CPTP Block	1177	838	2
Descriptors Register	98	72	-
Command Controller	193	174	-
Packet Formatter	199	120	-
Packet Decoder	252	147	-
Packet handler	227	125	-
CPTP CRC	26	17	-
CPTP PEC	77	18	-
CPTP Rx FIFO	81	102	1
CPTP Tx FIFO	29	26	1

TABLE X. CPTP, PROASIC3 RESULTS

	CPTP, Max packet size 64K, Max Tx/Rx descriptors 16, Max DMA length 128 Bytes, ProASIC3 target		
	Core Cells	***Registers***	***BRAMs***
CPTP Block	3989	869	4
Descriptors Register	396	76	-
Command Controller	857	184	-
Packet Formatter	746	152	-
Packet Decoder	793	136	-
Packet handler	820	135	-
CPTP CRC	86	18	-
CPTP PEC	218	34	-
CPTP Rx FIFO	337	104	2
CPTP Tx FIFO	140	28	2

TABLE XI. OVERALL NODE IP, VIRTEX 5 RESULTS

	Overall Core, Virtex 5 FX target		
	LUTs	***Registers***	***BRAMs***
Without Switch	6924	6596	10
With 2 external ports	12212	10837	19
With 6 external ports	16389	14098	19
With 14 external ports	24737	20954	19
With 30 external ports	50011	37471	19

TABLE XII. OVERALL NODE IP, PROASIC3 RESULTS

	Overall Core, ProASIC3 target		
	Core Cells	***Registers***	***BRAMs***
Without Switch	32478	7215	22
With 2 external ports	59411	13434	35
With 6 external ports	75781	17219	47

V. CONCLUSIONS

The designed and developed IP offers fine configurability options to the user, not only at top level, but offers also configurability of its constituents, allowing the user to tailor it according to the performance and utilization needs. Synthesis results on Virtex 5 and ProASIC3 have shown that the design fits the targeted devices.

Preliminary synthesis on RTAX2000 has shown that the Node IP without SpW Switch can also fit within the device as well as an implementation with 2 external ports if SpW Interrupts are not supported. The results have shown that the SpW Interrupt block in the Switch occupies a significant portion of the device resources, since the implementation (taken as is from a previous ESA study) implements all timers in registers instead of the LUT/memory based implementation followed for the End Node.

Node's performances meet the set requirements for both the Virtex and ProASIC3 implementations. The RMAP target response time of 1.2 us on Virtex 5 (including two times the SpW Switching latency) exceeds by far the requirements for SpW-D operation. Experimentation at much lower system clock frequency (20 MHz on the ProASIC3 implementation) resulted in a response time of 7 us showing that the performance requirements can still be met even at low system clock frequencies.

REFERENCES

[1] ECSS-E-ST-50-12C: SpaceWire-Links, nodes, routers and networks, 31 July 2008

[2] ECSS-E-ST_50-52C: SpaceWire – Remote Memory Access Protocol, 5 February 2010

[3] ECSS-E-ST_50-53C: SpaceWire – CCSDS packet transfer protocol, 5 February 2010

[4] ECSS-E-70-41A: Ground Systems and Operations – Telemetry and telecommand packet utilization, 30 January 2003

[5] ECSS-E-ST-50-54 Draft 1.7: SpaceWire Network Discovery and Configuration Protocol, version, 21 May 2015.

[6] SpaceWire – Time Distribution Protocol, VHDL IP Core's user manual. Version 1.1, July 2014 Aeroflex-Gaisler (https://amstel.estec.esa.int/tecedm/ipcores/SPWTDP_um.pdf).

[7] High accuracy Time Synchronization over SpaceWire Networks, Version 1.1, 29 September 2012, Aeroflex-Gaisler (https://amstel.estec.esa.int/tecedm/ipcores/time_sync_protocol.pdf).

[8] ECSS-E-ST-50-12C Rev.1 DIR3: SpaceWire-Links, nodes, routers and networks, 23 November 2015

[9] AMBA specification Rev 2.0, ARM 1999

Networks & Protocols 2 (Long)

A Graphical Method to Configure SpaceWire Networks

SpaceWire Networks and Protocols, Long Paper

Thomas Bahls and Alin O. Albu-Schäffer
Institute of Robotics and Mechatronics
German Aerospace Center (DLR)
Münchnerstr. 20 82234 Weßling, Germany
Thomas.Bahls@dlr.de

***Abstract*— Complex robotic systems like the DLR Hand Arm System integrates a huge amount of sensors and actuators. Hence system design and especially communication infrastructure design has to be flexible in a heterogeneous network of different bus systems. As basis, a modular electronic concept as well as a well-structured communication concept is necessary [1]-[3]. SpaceWire suites well to these requirements since on one hand it supports arbitrary topologies from point to point up complex network structures and on the other hand it is easy to implement and has a small footprint. Additionally its logical and regional addressing scheme enables changes in the topology during runtime simply by reprogramming the routing switches. However, such changes require expert knowledge. This work presents a graphical method to setup and configure SpaceWire network topologies. This enables non-experts to replace or integrate new components to the system or to set up a test bed to investigate a specific aspect. The developer provides a GraphML description [4] specifying the SpaceWire communication capabilities of each component. Thus the user is able to adapt the SpaceWire network topology or to set up a new one simply by merging the different GraphML descriptions of the used components. A post process is afterwards used to analyze the GraphML description and to generate the necessary configuration messages according to the topology. This enables faster development cycles and rapid prototyping. The approach is approved and explained using the SpaceWire network topology of the DLR Hand Arm System.**

***Index Terms*— Graphical Communication Infrastructure Design, Automatic Configuration, GraphML**

I. Introduction

The DLR Hand Arm System is an anthropomorphic impact tolerant robot based on variable stiffness actuators. It is designed to meet the human archetype respective to size, weight and performance (see Fig. 1). Its communication infrastructure comprises 52 actuators and 430 sensors of different types [2], [3]. To guarantee main control loops up to 10kHz a deterministic behavior, low latency, high bandwidth and mechanisms to synchronize the actuators and sensors are indispensable. Furthermore their arbitrary physical interfaces (I2C, SPI, BiSS, PWM) demands flexibility and modularity in terms of electronic and communication [1]-[3].

Fig. 1. DLR Hand Arm System light weight robot

SpaceWire provides this flexible communication infrastructure, since it enables arbitrary network topologies and is changeable during runtime. It also supports high speed communication up to 1Gb/s by use of an adapted physical layer [5] and is deterministic for a given topology. Another important aspect is the synchronization, of the participants of a SpaceWire network via timecodes.

This flexibility also has to be introduced to the hardware abstraction layer to enable the user to change, adapt and expand the topology according to his purpose. Changes in

existing systems as well as rapid prototyping of new systems with hardware out of the shelf can be performed. This flexibility was, however, not given so far as a standard user. All changes and adaptions to the system or simply the setup of a new testbed require a detailed SpaceWire knowledge.

A huge amount of publications exists dealing with different aspects of network configuration: For example topology depending configuration algorithms [6], [7], automation aspects of network configuration [8], [9], configuration management [10], [11] and network configuration in virtualized environments [8], [12]. Most of the approaches are either detached from a specific communication system described on a higher level of abstraction [6], [11] or are describing their approaches on standard IP-based systems and services where hardware and tools are already available and IP-specific configuration details are abstracted or hidden [7], [8], [10]-[12]. But so far according to the authors' knowledge there is no approach neither universal to describe network topologies for various communication systems taking into account their specific configuration mechanisms nor SpaceWire specific.

This paper provides a method to graphically setup network topologies according to user requirements taking the example of SpaceWire. Since network routing can be described as a problem of graph theory [13], a graph description of network topologies is proposed. Hence, the topology description is stored in the GraphML format (a XML based description of graph structures) [4] which is used in a post process to configure the network (routing and connections).

Section II gives a brief overview of SpaceWire and the DLR protocol suite which is built on top of the standard SpaceWire stack as well as a brief introduction to GraphML. Section III deals with the mapping of SpaceWire characteristics into GraphML attributes. The use of this mapping in the post process is explained in section IV as well as its application to the DLR Hand Arm System. Section V concludes the paper and gives a brief outlook to future fields of applications of the presented method.

II. BASICS

This chapter gives a brief overview of SpaceWire focused on its addressing scheme and the associated configuration. Afterwards a short introduction to GraphML is given by using a SpaceWire network as an example.

A. SpaceWire

The main intention of SpaceWire is the data exchange between sensors, processing units, mass-memory units and downlink telemetry subsystems onboard of a spacecraft [14]. Since SpaceWire's properties also meet the requirements of a robotic system (high speed, low latency, synchronization etc.) various systems developed in our institute use SpaceWire as communication backbone (e.g DLR HAND II, DLR Crawler, DLR Miro, DLR Mica, DLR Hand Arm System [15]-[18], [1], [3]). The following brief overview of the SpaceWire network layer is based on [14] and deals with terms and definitions of SpaceWire links, nodes, routing switches and networks. Furthermore the different addressing schemes are presented.

A SpaceWire network consists of an arbitrary number of three components:

- Nodes are start or end points of a SpaceWire network. They are connected via links directly to another node or to a routing switch.
- Routing Switches connect various nodes together via links and also provide routing capabilities. The maximal number of connections is limited to 31.
- Links provide the capability to exchange packets between nodes and routing switches in any combination (node ↔ routing switch, node ↔ node and routing switch ↔ routing switch).

Each packet to exchange has the following structure <destination address><cargo><end of packet marker>. The destination address describes the destination to route the packet to. Depending on the addressing scheme and the network topology the length varies from 1 to n bytes. The cargo contains an arbitrary number of bytes. The end of packet marker can be either a normal end of packet (EOP) or an error end of packet (EEP). The SpaceWire standard does not dictate a maximum packet length. Nevertheless, to achieve a deterministic behavior in our systems a limitation is necessary (our implementation limits the packet size to 1024 Bytes).

SpaceWire offers three different ways of describing the destination address. The first addressing scheme is "path addressing". Here the address section of the packets contains the physical path through the network. For example an address of <4, 4> routes the packet via port 4 of the first routing switch to a node connected to port 4 of the second routing switch. In "path addressing" the routing switch always performs header deletion which is necessary for the correct path through the network. In "path addressing" the number of bytes is arbitrary but the address range is limited to the range from 1 to 31 which represents the number of possible connections to a router. Address 0 is a special case used for configuration of the lookup table of the routing switch.

The second scheme is "logical addressing". Here the address section of the packet contains exactly one byte in the range of 32 to 254 (255 is reserved for future use). The route through the network is determined by the lookup tables of the routing switches. The lookup table can be preconfigured or configured at startup. Header deletion is optional but makes only sense when reaching the final link to the destination node.

The third addressing scheme is "regional addressing". Here the destination address contains an arbitrary number of bytes in the range of 32 to 254. This scheme is used if a packet has to be exchanged between different regions. Regions are necessary if the logical address space from 32 to 254 is insufficient or if a systematic separation makes sense. Each logical address except the last one represents a gateway to another region. Due to this leaving a region always requires header deletion of the actual first address. Finally in the last region there is no difference to any other packet using “logical address” in between this region.

Since all three addressing schemes can be used in parallel in SpaceWire networks, preferring "logical" and "regional

addressing" makes sense to keep the addressing overhead as small as possible. In our implementation path addressing is exclusively used for configuration of the lookup tables of the routing switches.

The SpaceWire standard does not define a transport layer. Since the control of a robot demands for reliable (connection oriented, request/response) and non-reliable (connectionless, datagram) mechanisms, a transport protocol was developed [3]. It is built on top of the existing SpaceWire packet and network layer. The transport protocol is wrapped in the cargo of a normal SpaceWire packet and uses the SpaceWire's address schemes. Since participants in a SpaceWire network have no knowledge about the network topology the transport protocol includes a configuration process. Here the destination address of the peer node is set. Therefore two specific nodes are able to set up a channel to exchange data by using the transport protocol.

B. GraphML

GraphML (Graph Markup Language) is a XML based format to describe graph structures [4]. The language provides core elements to fully describe graph structures enhanced by a mechanism to store graph independent application specific data [4]. Due to this, Brandes et al. [4] especially attach importance to the following points:

- Simplicity to easily be parsed and interpreted by humans and machines.
- Generality to support arbitrary graph models.
- Extensibility to provide additional information for arbitrary applications in a well-defined way.
- Robustness to easily extract additional data by any target application without the need of understanding or interpreting the whole graph model.

The following GraphML listing (see listing 1) shows the core elements of the language. The header part is omitted for reasons of clarity and comprehensibility (a detailed description is given at http://graphml.graphdrawing.org). The example consists of three SpaceWire nodes connected to one SpaceWire routing switch. Since GraphML only differentiates between nodes and edges, SpaceWire nodes as well as routing switch are modeled as GraphML nodes (`<node><node/>`). The feature of integrating additional data into the graph data is used to specify the type of SpaceWire item. By use the GraphML key `<key attr.name="type" attr.type="string" for="node" id="d0">` (see line 3 and 4) a data attribute of type string with identifier d0 is defined for the GraphML node element. By `<default>node/routingSwitch</default>` the default values are set here node or routingSwitch (see line 5). Inside a node statement the defined data can be set by `<data key="d0">routingSwitch<data/>.`

In SpaceWire the connection between nodes and routing switches are called links. The equivalent to a link in GraphML is the edge statement (`<edge></edge>`). By setting the source and the target element to one of the nodes unique identifier `<edge id="e0" source="n0" target="n1" />` the connection between the GraphML nodes can be determined.

```
<graphml>

  <key attr.name="type" attr.type="string"
   for="node" id="d0">
  <default>node/routingSwitch</default>

  <graph edgedefault="undirected">
  <desc>GraphML description of a simple
        SpaceWire network</desc>

    <node id="n0">
      <data key="d0">routingSwitch<data/>
    <node/>
    <node id="n1"/>
      <data key="d0">node<data/>
    <node/>
    <node id="n2"/>
      <data key="d0">node<data/>
    <node/>
    <node id="n3"/>
      <data key="d0">node<data/>
    <node/>

    <edge id="e0" source="n0" target="n1" />
    <edge id="e1" source="n0" target="n2" />
    <edge id="e2" source="n0" target="n3" />

  </graph>

</graphml>
```

Listing 1. GraphML listing of a simple spaceWire network

III. PEPRESENTATION AND SPECIFICATION

Chapter II showes that a SpaceWire topology can be described by the node and edge elements of GraphML. However, SpaceWire specific characteristics can only be added by integrating additional information (e.g. type of SpaceWire item: Node or routing switch). So a fully automatic configuration process of SpaceWire networks according to the standard as well as the transport protocol build on top is possible. To achieve this, the following data set of SpaceWire characteristics is integrated into the GraphML format.

a) type: (is assigned to the GraphML `node` tag) It specifies the kind of SpaceWire item. It could be one of the following items: **`node/routingSwitch/link/subnet`**. In addition to the already known SpaceWire items node and routingSwitch also link and subnet are provided. Link is a special case of the normal SpaceWire links which are described by `edge`. It represents a connection of two SpaceWire items (node or routingSwitch) via an exchange level implementation of the SpaceWire standard, which is necessary to connect physical separated parts of a SpaceWire network each with independent clock domains. It is independent of the configuration mechanism but already included for the future goal of using GraphML description as a base for code generation. Subnet describes the SpaceWire regional address scheme. Each item in a region is encapsulated by a node of the type subnet using GraphML nested graph description for hierarchical ordered nodes [4].

b) numberOfPorts: (is assigned to the GraphML `node` tag) Specifies the number of ports in case of a routingSwitch item. It could be in a range of 1 to 31. The default value is zero

which is used for all other SpaceWire items except the routingSwitch.

c) logicalAddress: (is assigned to the GraphML `node` tag) Specifies the SpaceWire logical address in case of a node or the regional address in case of a subnet. It could be in a range of 32 to 254. The default value is 32. It is ignored for routingSwitch and link items.

d) peerNodeAddress: (is assigned to the GraphML `node` tag) Specifies the logical address or regional address of a peer node for the transport level implementation, and could be in a range of 32 to 254. The default value is 33. It is ignored for all other items except a SpaceWire node.

e) port: (is assigned to the GraphML `edge` tag) Specifies the port on which edge is connected to a routingSwitch.

Furthermore, some additional data are helpful to improve the readability as well as for the future goal of using GraphML description as a base for code generation.

f) subsystem: (is assigned to the GraphML `node` tag) Specifies the physical subsystem of the item to show its membership for example in case of robot "joint1". That could also be useful for the application level to identify functional groups (e.g. "housekeeping" to read out all sensors of this group).

g) Label: (is assigned to the GraphML `node` tag) Specifies the name of an item. For example "I2CTemperatureBridge".

IV. Methods and Examples

To enable a configuration of SpaceWire networks in an automatic process a GraphML description of subcomponents (e.g. PCBs) according to the specification presented in chapter III is required. Therefore, every component which can be integrated into a SpaceWire network must provide a GraphML description of its interconnection possibilities. Starting from this the user can modify the description according to his requirements within the function range of the component. By putting all GraphML descriptions of the various components together the whole network topology can be set up.

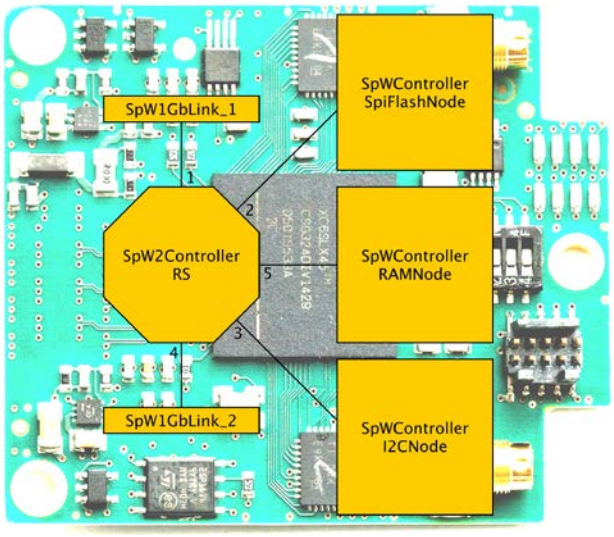


Fig. 2. FPGA based SpaceWire interfacing electronic overlayed with its graphical GraphML representation.

Since one of the design goals of developing GraphML was simplicity in terms of interpretation and parsing [4], the GraphML file describing a component can easily be written by hand. A much more practical solution is to use a graphical editor as yEd (https://www.yworks.com/). yEd provides full GraphML language support including the feature of integrating additional data. The graphical solution even makes the learning of the GraphML syntax needless and for the user the configuration process is reduced to assembling various predefined graphical components.

The depicted electronic component (Fig. 2). shows the SpaceWire-to-controller PCB as well as its associated GraphML graphical representation described with yEd. The PCB uses FPGA technology to implement the SpaceWire stack and the transport level build on top. Its purpose is to connect arbitrary standard buses (I2C, SPI, SSI etc.) or non-standard buses (processor bus, PWM, etc.) to the SpaceWire network. The data exchange is carried out by a dual-clock, dual-ported RAM interface (port 5 SpWControllerRAMNode in graphical representation). Additionally, it offers two 1Gb/s SpaceWire links for external connections (connected to port 1 and 4). Furthermore it also provides housekeeping (temperature, current measurement) functionality via I2C (port 3 SpWControllerI2CNode) and access to a SPI-Flash to reprogram the FPGAs content during runtime (port 2 SpWControllerSpiFlashNode).

This initial graphical description which provides all available features of the PCB can be adapted according to application specific requirements. Thus, a whole SpaceWire network can be set up by merging different components together into one single graphical description. The so generated final GraphML description serves as the starting point for the post process. The post process is implemented in python and contains the following steps:

1. The GraphML file is parsed to discover all nodes (`<node><node/>`). Their unique id as well as the embedded SpaceWire specific data are extracted to python dictionaries and saved in a list. The edges (`<edge></edge>`)are proceeded in the same manner.
2. All SpaceWire routing switches are filtered out of that generated GraphML node list by interpreting the SpaceWire specific data. Afterwards the unique GraphML node id of each discovered SpaceWire routing switch is used to search for matches in the source/target items of the generated GraphML edge list. Thus the GraphML node id and the port number of the counterparts connected to each SpaceWire routing switch can be extracted and saved in separate list associated to each SpaceWire routing switch. In this way all SpaceWire routing switches know their locally connected items.
3. Each SpaceWire routing switch's list of locally connected items is analyzed and in case of a SpaceWire node extended by the associated logical address. This is done by a repeated iteration through the GraphML node list to discover all unique GraphML node ids of the locally connected items. In

this manner a local LUT (lookup table) of logical address to port mapping is built.

4. All SpaceWire routing switches exchange their routing information (LUT of logical address to port mapping) to set up a global LUT associated to each routing switch. The used mechanism is similar to the reliable flooding process described in [13] with the difference that the network topology is known. The starting point of each global LUT associated to a SpaceWire routing switch is its own local routing table. This global LUT is exchanged among all SpaceWire routing switches found in the formerly generated list of all locally connected items of each routing switch. While doing that the received logical address to port mapping of the received global LUT has always to be changed in: logical address / port of the SpaceWire routing switch where the global LUT comes from. This has to be done for about n-1 times where n is the number of SpaceWire routing switches in the network. Since in a worst case scenario the topology of the connected SpaceWire routing switches is a line structure where each routing switch is connected to only two bordering routing switches.
5. The starting point to build SpaceWire configuration packets for set up the LUT in each SpaceWire routing switch is a specific node within the SpaceWire network. With the unique GraphML node id of this SpaceWire node the directly connected SpaceWire routing switch can be found inside the former generated GraphML node list. The only addressing scheme which can be used to set up the LUT inside the SpaceWire routing switches is path addressing. Due to this for the directly connected SpaceWire routing switch the destination address equals zero for accessing its internal LUT. Based on the locally connected items list of this SpaceWire routing switch directly connected routing switches and their associated port number are searched for. Their destination address is built by simply inserting the outgoing port number at initial position of the destination address of the previous SpaceWire routing switch. This has to be done in each new stage of connected SpaceWire routing switches until all routing switches are reached.
6. Since all SpaceWire routing switches of the network are configured at this point the logical addressing scheme can be used for the transport protocol built on top of the SpaceWire stack. By discovering all SpaceWire nodes out of the former generated GraphML node list their logical address and peer node address can be extracted which are necessary to build the transport protocol configuration packet. Since logical addressing is used the transport protocol configuration is independent from the SpaceWire node which performs the configuration.

The post process also supports the handling of different SpaceWire networks and the associated regional addressing scheme. Here, the exchange of the LUT is performed in each subnet separately. An adjacent subnet is treated as a normal SpaceWire node with a logical address. The GraphML nested graph approach is used for describing hierarchical graphs [4]. The unique id of each GraphML element is extended by a leading id describing the parent node (`<node id="n0::n1"><node/>`). Hence, it is simple to identify the membership of each SpaceWire item to a subnet.

The presented method is applied to the DLR Hand Arm System's SpaceWire network topology. As an example the SpaceWire topology of the HASy hand is used. The GraphML description of the SpaceWire topology is built by use of the graphical editor yEd (see Fig. 3).

The SpaceWire topology of the HASy hand is spread over five FPGAs located on three different PCBs. This is recognizable in the five vertical branches which have their orign at the horizontal structure at the bottom (see Fig. 3). The horizontal structure at the bottom is the backbone which connects all physically separated parts of the SpaceWire network. The depicted SpaceWire topology comprises in total 61 SpaceWire nodes (square shape) and seven SpaceWire routing switches (octagon shape). 61 SpaceWire nodes correspond to 61 logical addresses to be set up in the seven routing switches' LUTs. This leads to 427 configuration packets exclusively to use the logical address scheme. This is exactly the result of the python post process of the GraphML representation of the hand's SpaceWire topology. The number of SpaceWire packets necessary for the configuration of the transport protocol built on top is variable since some nodes share their peer nodes. In our case 14 nodes are unused, since they are reserved for optimizing latency by efficient packing. From the remaining 47 nodes one is used as a shared peer node for all others. This leads to 46 configuration packets associated with the transport protocol which are also the result of the python post process.

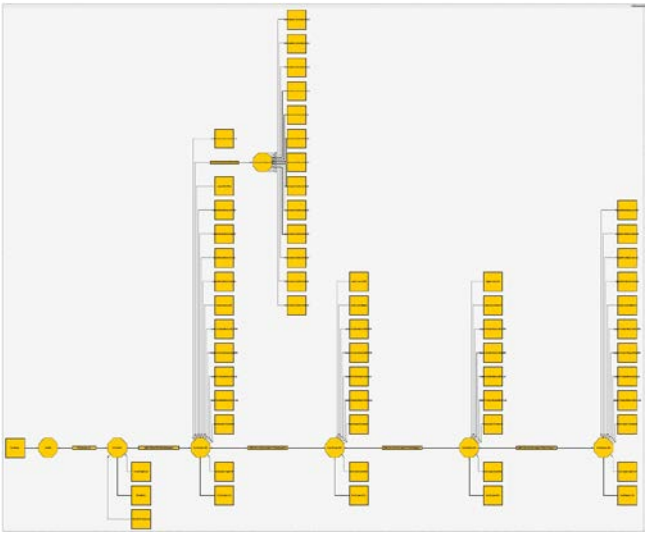


Fig. 3. DLR Hand Arm System's SpaceWire network topology of the hand

The additional data embedded in the GraphML format is not only limited to network and transport layer depending data. The post process also generates configuration files for our hardware abstraction layer approach "robotkernel" as well as our middleware approach "links and nodes". The "robotkernel" is a runtime-configurable robotic hardware abstraction framework. It is designed as a cross platform software component with reusable dynamical loadable device drivers encapsulated in modules. It provides an intra-module communication, a module synchronization mechanism and access to the cyclic and acyclic hardware data. It also supplies modules with generic interfaces to control applications, written in C, python, Simulink, etc. At DLR most of the robotic hardware components work with the "robotkernel" by simply writing a bunch of configuration files. As middleware between different processes we use our own hard-realtime capable middleware called "links and nodes". It provides a realtime communication mechanism for cyclic process data, acyclic service calls and process management across IP networks.

V. Conclusion and Outlook

This paper presents a graphical method for communication infrastructure design. Since network topologies can be described by graph theory the system's network topologies are modeled by using graphs. To store the graph information the XML based GraphML format is used. It is easy to read and interpret by humans and machines and provides a mechanism to store arbitrary additional data. Thus, specific information depending on the used communication standard as well as detailed information from other abstraction layers can be integrated. By use of a graphical editor (e.g. yEd) the usability can be further enhanced. By providing a graphical model of each component describing its network depending capabilities the system's topology can be adapted easily to user needs without expert knowledge. Complexity can be clearly described with the GraphML approach and detailed configuration knowledge can be hidden in the post processing stage. The method was applied to the SpaceWire topology used in the DLR Hand Arm System with promising results. The system can be fully configured by use of the method. Additional application specific information (also integrated in the GraphML format) can also be processed to configure the hardware abstraction layer.

Future work will include the modeling of all SpaceWire network components used in the DLR systems (e.g. DLR Miro, DLR Mica). Thus components can be used out of the shelf to easily build new testbeds or systems. Another possibility is to go a step further by using the GraphML description not only for configuration but also for code generation. Since a full SpaceWire stack as well as the transport protocol built on top and various communication bridges to physical interfaces (I2C, SPI, BiSS, PWM, etc.) are available in VHDL in our institute the whole component configurations could be generated online.

References

[1] M. Grebenstein, A. Albu-Schäffer, T. Bahls, M. Chalon, O. Eiberger et al., “The dlr hand arm system,” in Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011, pp. 3175–3182.

[2] S. Jörg, M. Nickl, A. Nothhelfer, T. Bahls, and G. Hirzinger, “The computing and communication architecture of the dlr hand arm system,” in Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011, pp. 1055–1062.

[3] M. Nickl, S. Jörg, T. Bahls, A. Nothhelfer, and S. Strasser, “Spacewire, backbone for humanoid robotic systems,” in Proceedings of the 4th International SpaceWire Conference, 2011, pp. 356–359.

[4] U. Brandes, M. Eiglsperger, I. Herman, M. Himsolt, and M. S. Marshall, “Graphml progress report structural layer proposal,” in Graph Drawing. Springer, 2002, pp. 501–512.

[5] M. Nickl, S. Jörg, T. Bahls and B. M. Cook “Towards high-speed spacewire links,” in Proceedings of the 5th International SpaceWire Conference, 2013, pp. 263–266.

[6] J. Kim and S. Lee, “Spanning tree based topology configuration for multiple-sink wireless sensor networks,” in Ubiquitous and Future Networks, 2009. ICUFN 2009. First International Conference on. IEEE, 2009, pp. 122–125.

[7] R. Emiliano and A. Mario, “Automatic network configuration in virtualized environment using gns3,” in Computer Science & Education (ICCSE 2015), Proceedings of 10th International Conference on. IEEE, 2015, pp. 25–30.

[8] S. Lee, T. Wong, and H. S. Kim, “To automate or not to automate: on the complexity of network configuration,” in Communications, 2008. ICC’08. IEEE International Conference on. IEEE, 2008, pp. 5726–5731.

[9] K. A. Weinstein, W. Wang, K. M. Peters, D. P. Gelman, and J. Dimarogonas, “A domain-level data model for automating network configuration,” in Military Communications Conference, 2010-MILCOM 2010. IEEE, 2010, pp. 1337–1342.

[10] K. Elbadawi and J. Yu, “Improving network services configuration management,” in Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on. IEEE, 2011, pp. 1–6.

[11] Z. Ismail, R. Hassan, A. Patel, and R. Razali, “A study of routing protocol for topology configuration management in mobile ad hoc network,” in Electrical Engineering and Informatics, 2009. ICEEI’09. International Conference on, vol. 2. IEEE, 2009, pp. 412–417.

[12] X. Ge, H. Jin, S. Wu, X. Shi, and W. Gao, “A method of multivm automatic network configuration,” in Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on, vol. 3. IEEE, 2009, pp. 309–313.

[13] L. L. Peterson and B. S. Davie, Computer networks: a systems approach 5th edition. Elsevier, 2012.

[14] E. Secretariat, “Spacewire-links, nodes, routers and networks,” ECSSE-ST-50-12C, Noordwijk, The Netherlands, July, Tech. Rep., 2008.

[15] S. Haidacher, J. Buttefass, M. Fischer, M. Grebenstein, K. Jöhl, et al., “DLR hand ii: Hard-and software architecture for information processing. In: Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on. IEEE, 2003. pp. 684-689.

[16] M. Görner, T. Wimböck, A. Baumann, M. Fuchs, T. Bahls, et al., “The dlr-crawler: A testbed for actively compliant hexapod walking based on the fingers of dlr-hand ii,” in Intelligent Robots and Systems, 2008. IROS 2008.

[17] U. Hagn, M. Nickl, S. Jörg, G. Passig, T. Bahls, et al., “The dlr miro: A versatile lightweight robot for surgical applications,” Industrial Robot: An International Journal, vol. 35, no. 4, pp. 324–336, August 2008.

[18] S. Thielmann, U. Seibold, R. Haslinger, G. Passig, T. Bahls, et al., “Mica - a new generation of versatile instruments in robotic surgery,” in IROS 2010, IEEE International Conference on Intelligent Robots and Systems, October 2010.

Multichannel Adaptive Routing for Intensive Data Packet Flows Transmission

SpaceWire networks and protocols, Long Paper

Elena Suvorova, Yuriy Sheynin, Valentin Olenev, Irina Lavrovskaya

Saint-Petersburg State University of Aerospace Instrumentation
Saint-Petersburg, Russia
{suvorova, sheynin}@aanet.ru, {valentin.olenev, irina.lavrovskaya}@guap.ru

***Abstract*—In many networks there is a necessity to transmit data packets flows, the intensity of which exceeds the throughput of one channel SpaceWire, GigaSpaceWire, SpaceFibre. This flow can be a packet flow from a single source to a single destination, for example, from a camera to a monitor. Also, a packet flow can include packets from different sources to different destinations that goes via two neighboring routers. An example is transmission of packets between two routers located on the boundaries of neighboring regions. The packets, belonging to one flow can have almost same length (transmission of uncompressed video), or quite different length (transmission of compressed video, transmission of packets with different content between two regions).**

The adaptive routing can be used for transmission of such packet flows. This mechanism includes in SpaceWire standard. A set of alternative output ports (a group of ports) can be determined in routing table for the logical (or regional address). Any output port from this group (if connection for this port is established and port is not occupied by other packet) may be used for transmission of packet with this address.

Thus, the summary throughput of all ports belongs to the group can be used for transmission of data packets with this address. However, the possibility of parallel transmission of packets from this flow to different output ports belongs to the group is required for effective utilization of this summary throughput.

If the length of packets may be different or if the quantity of input ports for considered flows is not equal to the quantity of output ports in the group, the router should include special mechanisms to ensure efficient parallel transmission of packets to all ports belongs to the group.

Adaptive routing for intensive data flows transmission can be implemented not only in SpaceWire/GigaSpaceWire networks, but also in SpaceFibre networks. We consider the specific of its implementation taking into account the features of data link layer (virtual channels with a fairly large buffers, retry mechanism)

In this paper we discuss possible implementations of these mechanisms for SpaceWire, GigaSpaceWire and SpaceFibre, estimate achievable bandwidth utilization of port's group, the overhead of the implementation of these mechanisms for packets flows with different characteristics.

A side effect of adaptive routing is a possible mismatch of the order in which packets are sent to the network form the source and the order of their receipt by destination. We evaluate the packet's window size that required in the destination node for recovery order of packets and the associated delays.

The ports of router belonging to the same group may be connected to one or several different routers (according to the standard). In the first case all packets from the flow will be transmitted via one chain of routers (via one path via network). In the second case, they will be transmitted through the network in different ways. The reordering of packets is possible in both cases. However, in the first case, the mechanisms, that prevent the packets reordering, can be implement in routers. But its implementation can lead to decrease of throughput utilization, to additional hardware costs and to increase of packet's transmission time. In the paper we estimate these overheads for data packets flows with different parameters.

***Index Terms*—SpaceWire, GigaSpaceWire, SpaceFibre, Adaptive routing**

I. Introduction

This paper discusses group adaptive routing mechanisms and their facilities in SpaceWire, SpaceFibre and GigaSpaceWire networks. We chose several case studies which find applications in different networks. The first case study considers high intensity data flow transmission (high resolution video) between source and destination. The second case discusses possibilities of several data flows transmission from different independent sources and destinations between two routers which are situated on borders of network regions. The third case study refers to group adaptive routing abilities for bypassing overloaded network parts. Finally, the fourth case study discusses group adaptive routing for bypassing failed links and routers.

II. SpaceWire Group Adaptive Routing

Group adaptive routing is defined in the SpaceWire standard [1,2]. In this switching mode, a routing table defines a set of output ports that correspond to one logical address. A packet with a particular logical address can be potentially passed to any of output ports in the defined group. Any port from the group with established connection and which is not occupied by transmission of another packet can be chosen for transmission of the subsequent packet. If all ports from the group are in the Run state and are occupied by transmission of other packets, the current packet will wait for one of these ports

to be free and ready for transmission. Output ports from one group can be connected either to the same or to different network devices. In the first case, packets transferred by means of group adaptive routing will be always transmitted via the same path i.e. via the same sequence of routers (see Fig. 1). In the latter case, packets will be transmitted via different paths, i.e. via different sequences of routers (see Fig. 2).

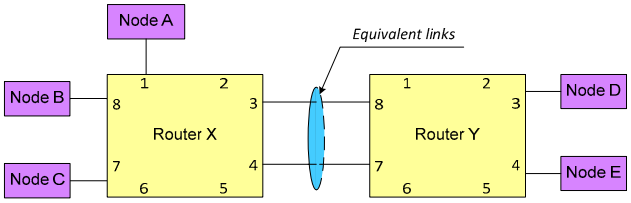


Fig. 1. An example of a network structure with group adaptive routing for connection with the same device

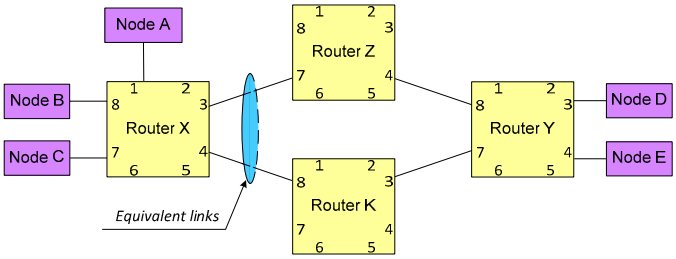


Fig. 2. An example of a network structure with group adaptive routing for connection with different devices

Group adaptive routing can be used [3, 4, 5, 6, 7, 8, 9, 10]:

- for co-utilisation of bandwidth of several links;
- for keeping a possibility to transmit data in case of link disconnection for one or several links.

In the next sections of the paper we consider several typical case studies of the group adaptive routing use and estimate its efficiency by the following criteria:

- Objective function, which is defined in the statement of the problem for each case study;
- Necessity and size of additional buffers;
- In-order packet transmission (if necessary).

Group adaptive routing can be also used in GigaSpaceWire [11] and SpaceFibre [12] networks, consequently, we will make estimation for these standards as well.

III. Case Study 1. High Intensity Data Flow Transmission between Source and Destination

A. Statement of the Problem

Let us consider the case when there are source and destination nodes in a network and data intensity between them is higher than the bandwidth of physical links between them. For example, this can occur for a source and destination of high resolution video traffic. From the structural point of view, it is possible to organize a path comprising several physical links. Examples of such paths are given in Fig. 3. At the logical level we can use the group adaptive routing for packets distribution between physical links. Packets in a data flow can have either of equal length (e.g. uncompressed video) or unequal length (e.g. compressed video). Such kind of applications may require in-order packets delivery. These nodes can exchange other types of traffic, e.g. command data, in addition to the high intensity traffic.

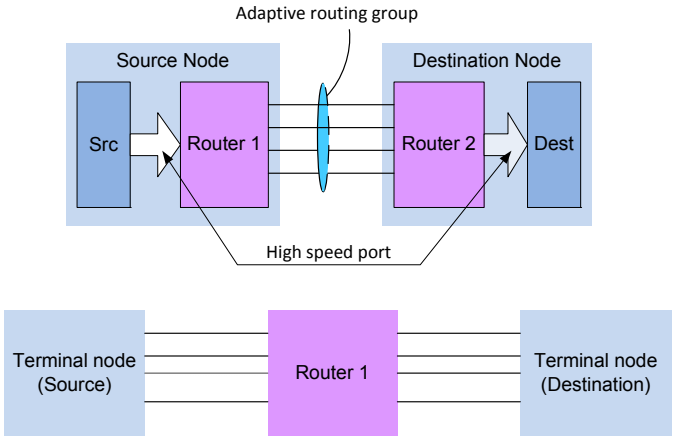


Fig. 3. Examples of terminal nodes connection for transmission of high intensity data traffic

Data generation rate in the source and data reception rate in the destination can be several times greater than data transmission rate over a single physical link.

B. Estimation of Characteristics for SpaceWire

Using of the group adaptive routing can lead to inefficient use of links bandwidth. This case can be shown for the receiver terminal node with embedded router with 4 external and one internal ports. Let us assume that the functional part of the receiver node can accept data 4 times faster than data rate in external ports of the terminal node. These external ports are used for reception of the traffic. If data packets are transmitted directly from external ports to the receiver, we will get the situation shown in Fig 4.

Data transmission between external ports and the functional part of the receiver is performed by packets. Until the current packet from the external port is not fully transmitted to the functional part via the internal port, the subsequent packet from another external port cannot start its transmission. Data character reception in the external port takes significantly more time than it takes for its transmission to the functional part. In the given example, there is a difference by 4 times. Consequently, there is a standby of a link to the functional part for a long time, which is ¾ of all operational time. In this case, group adaptive routing cannot increase the link bandwidth.

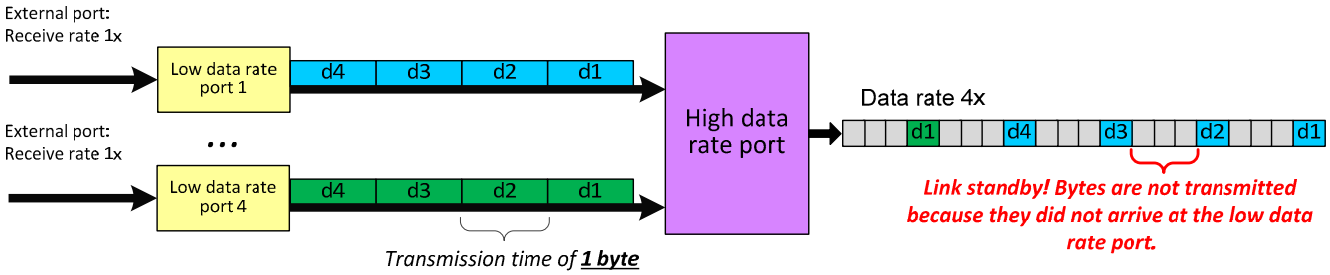


Fig. 4. Data transmission without buffering

This problem can be solved by adding buffers to the low data rate ports. The high data rate link's bandwidth can be fully utilized if a packet starts its transmission from the external port to the functional part just after it is fully stored in the buffer (see Fig. 5).

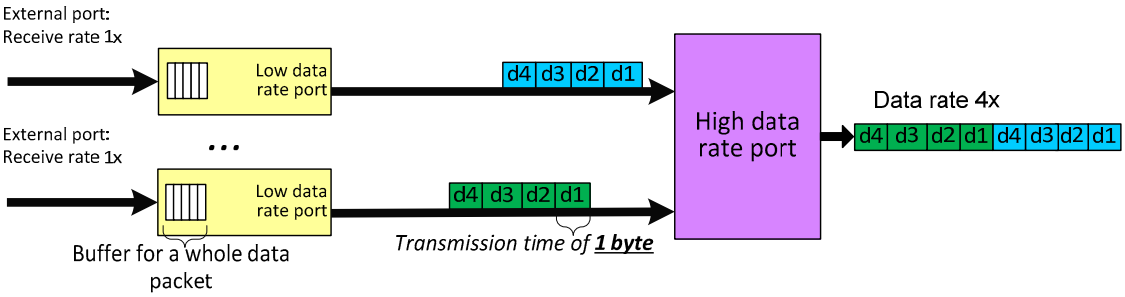


Fig. 5. Data transmission with buffering

The size of transmitted packets should be not more than the buffer size. Such a scheme with buffering requires an additional arbitration mode for data transfer to the high data rate port. An arbiter should choose a low data rate port for transmission in the following cases:

- A buffer of a low data rate port contains an entire packet;
- A buffer of a low data rate port is full (i.e. packet's length is more than the buffer's size).

If packet's length is more than the buffer's size, then the part of the packet that was not stored into the buffer will be transmitted to the functional part at a low data rate. This will result in decreasing of the link bandwidth.

For the reason described above, the transmitting terminal node should also implement buffers in its external ports. A generalized scheme which shows the place of buffers in receiver and transmitter is given in Fig. 6.

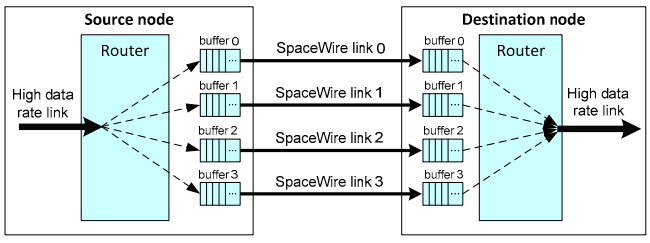


Fig. 6. Place of buffers in receiver and transmitter

However, this buffers mechanism can be insufficient. Packets flow can contain packets which have the size equal or less than the buffer's size. Consequently, the following situation can occur: the buffer contain a packet or its fragment and still has some free space. The next packet from the functional part can start to be stored in this buffer. And yet, there are no enough space in the buffer to store the whole packet. As a result, a part of this packet, which was stored to the buffer, is transmitted at a high data rate, while the other part is transmitted to the buffer at a low data rate of the external port. During this time, subsequent packets from the high data rate link cannot be transmitter to other external links. Therefore, there is a significant decrease of the high data rate link bandwidth.

As an example of a packets flow with different sizes let us consider a compressed video flow. Video frames compressed using MPEG, H.263, H.264 and other standards can vary by in sizes by several times (even by ten times) [13, 14]. Let us consider data transmission from a compressed video source with the following sequence of frames: IBPBPBPBP (see Fig. 7), where:

- I-frame – intra pictures;
- P-frame – corresponds to the frame compressed using a reference to one image (P - predicred);
- B-frame – corresponds to the frame compressed using a reference to two images (B - bidirection).

Fig. 7. Compressed video frames

In the discussed example a node has three external ports with buffers. Size of these buffers corresponds to the maximal size of I-frame in a system. A possible sequence of packets transmission is shown in Fig. 8. Dotted lines show buffers bounds.

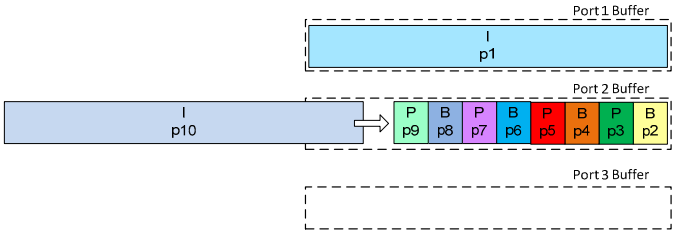


Fig. 8. Example of packets sequence transmission to output ports

The first packet with I-frame can be stored into the Port 1 buffer. This I-frame is followed by 4 groups of BP-frames. Each such frame is significantly smaller than the I-frame. Assume that they were stored into the Port 2 buffer and still left free space in this buffer. The next frame is an I-frame, which can be passed to any buffer:

- Port 1 has already send some of data and freed free space;
- Port 2 has free space because 4 BP-frames are smaller than the buffer size;
- Port 3 buffer is empty.

According to the rules of group adaptive routing a packet can be passed to any of these ports. Assume that it will be passed to the port 2 buffer. The bigger part of the packet could not be stored in the buffer so it will be stored in the buffer at a data rate of the external port. Consequently, the high data rate link bandwidth can decrease by 1,5-2 times depending on sizes of packets and port arbitration choice.

A similar problem can occur with packets of the same size which is equal the buffer size. For example, in a terminal node with three external ports the first packet was sent to the port 1 buffer, the second – to the port 2 buffer. At the time of starting transmission of the third packet, a part of the first packet was send, so there is again free space in the port 1 buffer. The port 3 buffer is empty but according to the rules of the group adaptive routing the third packet can be passed to the port 1.

In some applications it can be essential to preserve at reception the packets order the same to the order they were sent by the source. However group adaptive routing cannot guarantee in-order delivery. An example of packets reordering is shown in Fig. 9.

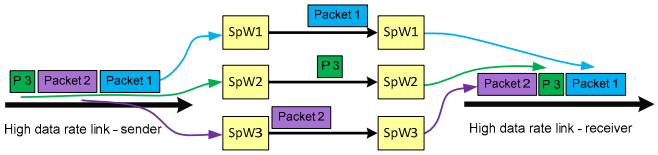


Fig. 9. Example of packets sequence transmission to output ports

In this case we did not loose the high data rate link bandwidth but the shorter packet P3 was fully stored in the receiver buffer before the longer one P2 was received. Therefore, the packet reordering occurred.

Link disconnection in one of the parallel links of the adaptive group can result in:

- loss of one or more packets;
- packets reordering while their transmission from high data rate link to the low data rate port.

These cases are illustrated in Fig. 10.

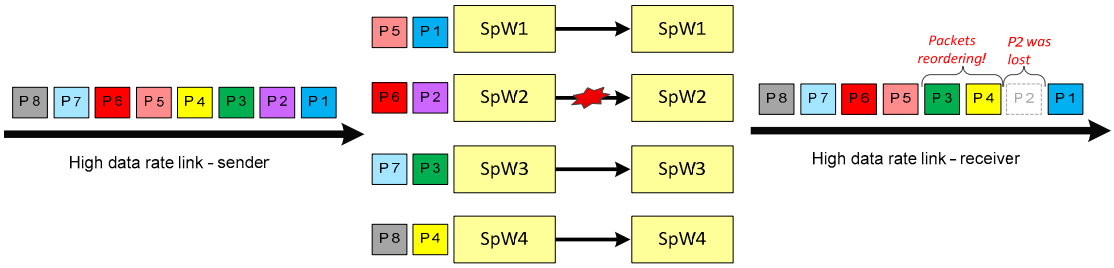


Fig. 10. Example of system behavior in case of link disconnection in one port

In this case we can observe a link disconnection in Port 3 which results in the loss of packet P2. The receiving side needs some time to detect the disconnection. Meanwhile, ports SpW3 and SpW4 can receive packets P3 and P4 correspondently. The order in which these packets will be passed to the high data rate link is random. Thus, the packets P3 and P4 can be passed in wrong order.

The more buffer size in ports is, the more data can be lost in case of link disconnection. According to the SpaceWire standard, the tail of a transmitted packet will be automatically spilled if a link disconnect occurs. If a transmit buffer contains other packets or their fragments, they will be:

- either transmitted in case of successful link reinitialisation;
- or they will be deleted in case of link disable flag is set to 1.

This time can be rather long (not less than 19,2 μs) and other ports can transmit a large number of packets. In the first case it can lead to significant reordering of packets on a sender side. Assume the situation when a packet was not fully written to the external port buffer because there was not enough free space and a link disconnection occurred. All next packets from the sender will be blocked for a long time (until the link is

reinitialized or link disables flag is set to 1) and could not be passed to other ports with established connection.

The discussed above cases show that smaller buffers can cause smaller losses of data and packet reordering in case of link disconnection. Disconnection results in loss of all data in transmit buffers together with packets' tails which were not stored in a buffer. However, buffer size should be defined in accordance with sizes of packets in a system. The smaller packets size is, the bigger transmission overhead is (see Fig. 11). The graphs of dependency between maximal packet size (header|overheads + data payload) and throughput utilization are represented in this figure. Here we can observe a contradiction between these requirements that should be balanced in practical engineering design.

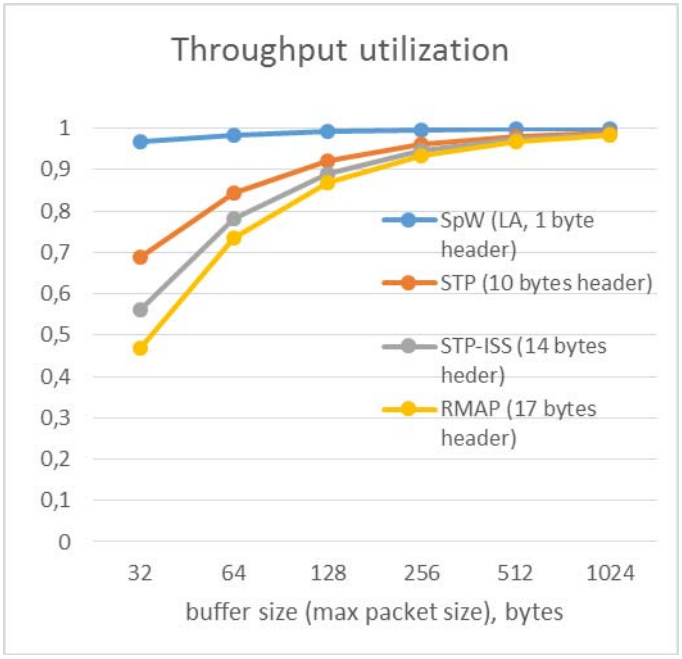


Fig. 11. Throughput utilization

Therefore, we can formulate the following summary:

- The objective achieving requires implementation of additional buffers, which size is not less than the size of transmitted packets, and an additional arbitration mechanism.
- The objective cannot be fully achieved even with these additional mechanisms: packets reordering and long delays are possible.

C. *Estimation of characteristics for GigaSpaceWire*

GigaSpaceWire standard has much in common with SpaceWire standard: layers over Exchange layer are the same, link disconnection and link re-initialization cause similar actions. Buffering in GigaSpaceWire is organized in the same way, as in SpaceWire. Consequently, if we use GigaSpaceWire in the first case study, we need additional buffering and arbitration mechanisms.

However, this also cannot solve all stated problems: packets reordering and long delays in some cases.

D. *Estimation of characteristics for SpaceFibre*

In contrast to the previous standards, in SpaceFibre there are output and input buffers (not less than 256 bytes) for each virtual channel and data retransmission in case of errors in a link. If these buffers are large enough to store an entire packet, than there is no need to use additional buffers for group adaptive routing.

Data retransmission mechanism allows significantly reduce data transmission delay in case of single errors in a link. However, the same problems as in SpaceWire arise if link disconnection lasts for a longer time: packets reordering and long delays.

IV. Case Study 2. Transmission of Big Data Between Different Senders and Initiators via Two Routers

A. *Statement of the problem*

A network can contain two routers which exchange an amount of data bigger than it can be transmitted via one physical link. For example, such routers can be situated on borders of regions. At a structural level such routers can be connected via several links in order to provide the required bandwidth. An example of such a network is given in Fig. 12. This pair of routers can transmit data packets flows between different pairs of sources and destinations. Packets flows can have different characteristics, such as different length of packets, different intensity. Data transmission rates in different ports can also vary. Generally, there is no need of preserving of the packets order between different packets flows in such adaptive routing application. However, in some cases it may be necessary to preserve an order for the packets between one pair of source and destination.

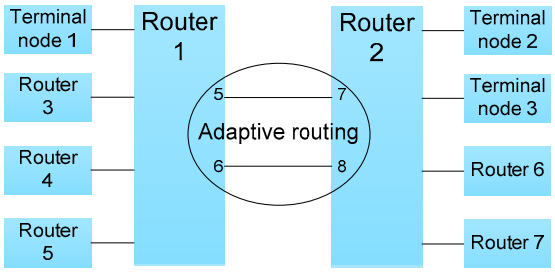


Fig. 12. An example of network structure

Assume that the system is designed correctly and an aggregate throughput of physical links between two discussed routers (hereinafter, adaptive group ports) is sufficient for transmission of all existing packets flows. In addition, the number if others ports in each of these routers can be equal or not to the number of ports in adaptive group ports. Data traffic coming via ports, which are not in the adaptive group, can be transferred either only to the adaptive group ports or not to this group. Transmission data rates can vary for different ports.

B. *Estimation of characteristics for SpaceWire*

As it has already been shown in the first case study, it is necessary to use additional buffers in low data rate ports if data rates vary in different ports.

If packets from several input ports can be transmitted to one output, then we can use buffers in the input ports in order to store the packets waiting for their transmission to an output port.

Let us consider the simplest case from the estimation characteristics point of view:

- number of ports in the adaptive group is equal to a number of ports which are not in the adaptive group;
- data rates are the same for all ports (buffers can not be used).

If there is a packet which should be transmitted to the adaptive group ports, then there always will be a free port in the group for transmission. If there is a packet from the adaptive group ports which should be transferred via a port not in the group, this port can be occupied by transmission of the other packet. In this case, packet transmission delay depends on the occupancy of an output port and packets length.

If transmission delay in one of output ports (includes in alternative group) occur, it does not affect to data flows (from other sources) that are transmitted via other ports

For example, if in the network shown in Fig. 12 delay occur for a packet transmitted from the router 3 to th router 6 via the port 5 of the router 1, it does not affect to transmission of data from the router 4 to the terminal node 2 via the port 6 of the router 1.

This can lead to much less decrease of link bandwidth in the adaptive group ports in contrast to the Case Study 1 (see Fig. 13). The example shows that Router 4 cannot be ready for some time to receive a data packet from Router 1 transmitted via a link from an adaptive group. During this time, data from the terminal node 1 can be transmitted to the terminal node 2 via another link between the routers without any loss of link bandwidth. However, if the terminal node 1 needs to send packet to the Router 3, then this packet and all subsequent packets will wait until the current packet is transmitted from Router 1.

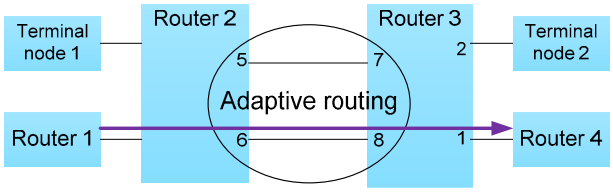


Fig. 13. Example of data transmission delay via adaptive group link

The same can be referred to a link disconnection case in one of alternative ports.

Let us consider a more complex case when a number of ports in the adaptive group is less than the number of ports out of this group. In this case, it is recommended to use buffers in order to store packets while waiting for output port is free. For example, assume that a router has 12 ports:

- 4 ports in the adaptive group;
- 8 ports can transmit packets addressed to the adaptive group.

Packets from 4 ports, which are not included to the adaptive group, can be transmitted to 4 ports from the adaptive group when a new packet arrives and needs to be passed to the adaptive group. This packet will wait until one of the ports from the adaptive group is free. If this packet does not fit in the buffer, then its tail will occupy one or several previous routers,

If data transmission rate is the same for all links then there is no need for additional buffers in output ports. As it was shown in Case Study 1, it allows to loose minimal amount of data in case of link disconnection in one of group adaptive ports.In this case, inefficient distribution of packets between ports and decrease bandwidth of adaptive group ports is impossible. Distribution of packets between the ports can influence transmission delays only.

However, if data transmission rates in ports vary, then we need additional buffers in low data rate ports. In this case, the problems from the Case Study 1 arise: decrease of link bandwidth in case of unsuccessful port arbitration, large amount of lost data in case of link disconnection.

Therefore, this case study requires additional buffers only in case of different data rated in links. The objective is successfully achieved.

The objective function is also successfully achieved in case of GigaSpaceWire and SpaceFibre use.

V. Case Study 3. Group Adaptive Routing for Bypassing Overloaded Network Parts

A. *Statement of the problem*

There are several paths for data transmission between the source and destination. Besides the discussed traffic, another traffic can be transmitted via these paths also. Transmission paths can be loaded in different ways at different moments of time. Potentially, group adaptive routing can be used for packets redirection to the less loaded path at a particular moment of time.

B. *Estimation of characteristics for SpaceWire*

Fig. 14 presents a fragment of a network structure with two alternative paths between the source and destination. Each alternative path comprises several routers, each of which can transmit additional traffic (see Fig. 14).

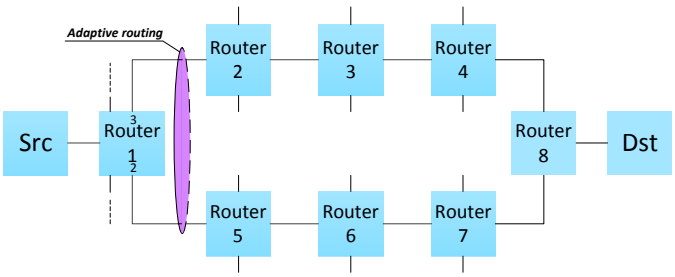


Fig. 14. Example of a network with two alternative paths

Packets from the source node Src arriving at Router 1 can be transmitted either to the Router 2 (path 1) or to the Router 5 (path 2) by means of group adaptive routing defined for ports 2 and 3. Router 1 arbitrates packet transmission from Src to the Router 2 via port 3 or to the Router 5 via port 2. If port 3 is occupied by transmission of another packet while port 2 is free, then the packet will be transmitted via port 2. If both ports are occupied, then the packet will wait them to become free.

If there is another traffic transferred to Router 5, then the discussed packet will be transmitted to the Router 5, if port 3 is not occupied. In this case group adaptive routing allows to effectively redirect packets to the free port.

However, this mechanism works only if an overload occurred in a router which divides further path into two alternatives. Assume that an overload occurred in Router 4 (see Fig 15) and, consequently, packets transmission from Src to Router 8 will be delayed. Although SpaceWire standard uses wormhole routing there are buffers of a particular size in routers. These can be credit buffers and any additional buffers.

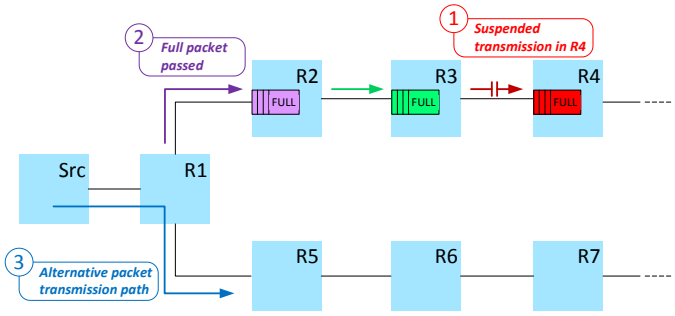


Fig. 15. Example of packet transmission via alternative path

Depending on packets length and buffers sizes the discussed packet can be either fully stored in Router 4 or its parts will be stores in Routers 3 and 2. In addition, if the Router 2 has still more space in buffers to accept the next packet, then Router 1 cannot predict that there is an overload somewhere further in this transmission path.

SpaceWire standard does not define any mechanisms for monitoring such kind of overloads in remote routers. SpaceWire flow control mechanism operate in a data link and can be used for detection of overloads in neighbor routers only. If there are no FCTs received, then ports can be considered as overloaded.

As a result, several subsequent packets can be transferred to Router 2 until all buffers on a transmission path are full. Consequently, an overload area will reach Router 1. There are two possible cases:

1. the next packet will be fully transferred to Router 2;
2. the next packet will be partly transferred to Router 2.

In the first case next packets from Src can be transferred to Router 5 (see Fig 15).

Packets transmitted via alternative path can arrive at Destination earlier than packets which were blocked while transferrin through Router 4. If there is a need of in-order packets delivery, then the destination node should have big enough memory in order to recover an initial packets order at a Transport and Application layers. The memory area should not be less than total buffers sizes in each existing alternative path between the source and destination.

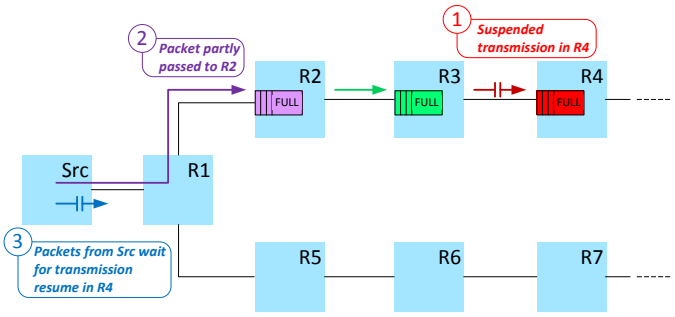


Fig. 16. Example of packets waiting for data transmission recovery in Router 4

In the second case, packets transmission from the source in Router 1 will be stopped until data transfer through Router 4 is recovered. This is shown in Fig. 16.

New revision of the SpaceWire standard proposes a mechanism of port time-outs in routers in order to detect a packet that has become stuck. This time-out controls the time since the last data character was sent from the input port to the output port. This mechanism can be used for discarding packets which have become stuck. However, it cannot improve anything in packets delivery avoiding overloaded parts of network.

This case shows that group adaptive routing use combined with such network structure does not give ability to dynamically control overloads and redirect packets from overloaded parts of network. This is because each router knows only about ports overloads in neighbor routers while it does not anything about remote routers n a network. This can result in inefficient use of the group adaptive routing is adaptive paths includes several routers.

Let us consider a subcase with branching adaptive paths in each router. Each router can choose one of alternative directions in dependence on neighbor routers state. An example of such network structure is given in Fig. 17.

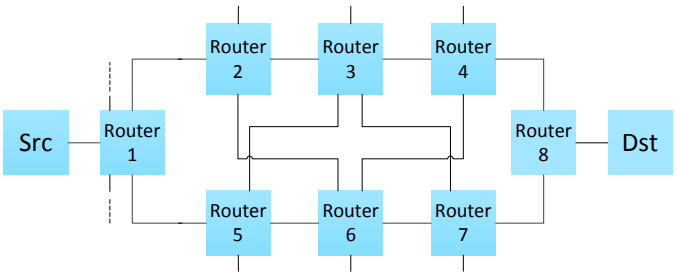


Fig. 17. Example of a network with adaptive connections between all routers in adaptive paths.

In this example, a packet in each router can be transferred via two paths. For example, a packet from Router 2 can be transferred to Router 3 or to Router 6. Let us consider the case when a packet from source to destination becomes stuck in

Router 4 as a result of an overload. Similarly to the previous case, two cases are possible:

1. A packet was fully sent by one of the previous routers;
2. Part of a packet was not sent.

In the first case all subsequent packets can be transmitted bypassing an overloaded network part. In the second case, the tail of unsent packet will block this opportunity. Therefore, the problem will be solved partly. According to SpaceWire standard, it is impossible to completely eliminate a possibility of keeping a packet tail in a router. As it was mentioned above, the only mechanism that can help in overload detection is a flow control in SpaceWire. Flow control is performed in terms of flits, which do not correlate with packets sizes. One flit can contain parts of several packets. Moreover, packet is unlimited in size. Even though it is limited and conforms to buffers in routers, the problem cannot be fully solved.

Moreover, it should be mentioned, that such network design causes a large number of additional communication paths between routers and, consequently, additional router ports which are necessary for adaptive path construction. This, in turn, will lead to increase of number of routers in the network.

The objective function is reached without additional mechanisms in routers, but alternative data path via each router should be implemented for this sample. Data packets sequence between the source and the destination nodes can be reordered in the network.

C. Estimation of characteristics for GigaSpaceWire

A GigaSpaceWire router likewise a SpaceWire router can determine load of only neighbour routers (using of the credit mechanism). Therefore all problems of SpaceWire standard shown for this use case are actual for GigaSpaceWire also.

D. Estimation of characteristics for SpaceFibre

A SpaceFibre router also can determine the state of only neighbour routers. It follows same problems as for SpaceWire and GigaSpaceWire standards. Due buffers of large size used in the data link layer of this standard, the overload of the neighbour routers will be determined very late, and more data will be stalled in buffers. Therefore data packets flow reordering will be more essential in comparisson with SpaceWire and GigaSpaceWire standard.

VI. Case Study 4. Group Adaptive Routing for Bypassing failed Network Parts

A. Statement of the problem

Some devices and communication lines can be failed in the network operation. The network can include spare routers and interconnection lines. Group adaptive routing can potentially be used for automatic (witout reconfiguration of the network) redirection of data packets flows bypassing failure equipment.

B. Estimation of characteristics for SpaceWire

The SpaceWire standard does not support guaranteed packets delivery. If failure occur in an interconnection line when data packet is transmitted, only the primary part of the packet (transmitted before the failure) with EEP will be forvarded farther via network. The rare part of the packet will be lost.

Every SpaceWire router can determine disconnections in its ports (in lines, connected this router with neighbours). Failure in a router that be accompanied with disconnection looks like link failure with this router for its neighbours.

A router has not any information about state of others (not neighbour) routers and interconnections. Therefore, the group adaptive routing is not suitable for network structure, shown in Fig. 14.

For example, if disconnection error occur between the router 4 and the router 8, the secuence of packets that wait for transmission will be placed in the routers 4, 3, 2. These packets will be never transmitted to destination if connectionbetween the router 4 and the router 8 will not restore.

As indicated above, if in the router 1 stalls the rare part of a packet, it will block transmission of next packets for long time. (These packets will be not transmitted via the router 5 until the rare part of the previous packets block the router 1.)

The mechanism of transmission timeouts in the SpaceWire standard next version allows to solve this problem by deletion of the untransmitted rare part of the packet in case of timeout.

But next packets can be transmitted to the router 2 (not to router 5) due to connection with router 2 is valid (available). These packets will not reach the destination due to disconnection between the router 4 and the router 8.

More suitable for this use case is network structure represented in Fig. 17. If in this structure disconnection between two routers occurs, the alternative path (that exist in every router) will be used for data packets transmission. For example if disconnection between the router3 and the router 4 occurs, next packets will be transmitted from the router 3 to the router 7, bypass failure connection. There are three alternative pathes between source and destination in this network:

(1) the router 1 – the router 2 – the router 3 - the router 6 – the router 7 – the router 8

(2) the router 1 – the router 2 - the router 6 – the router 7 – the router 8

(3) the router 1 – the router 5 – the router 6 – the router 7 – the router 8

Packet order is not guaranteed when packets transmited via different pathes.

So for this use case the objective is reached without any additional mechanisms in routers, but alternative data pathes should be determined for every router.

C. Estimation of characteristics for GigaSpaceWire

GigaSpaceWire router likevise SpaceWire router has information only about state of connections with neighbour routers. Correspondingly, all problems shown for this use case when SpaceWire standard is used are actual for GigaSpaceWire standard.

D. Estimation of characteristics for SpaceFibre

A SpaceFibre router also can determine the state of connections with neighbour routers only. On data link layer of SpaceFibre standard the data frames retransmission is

implemented. The quantity of retransmission or retransmission time is unconstrained. Therefore for detection of permanent failures we need to add retransmission timeout. If this timeout expired, the port (and the data link) will be marked as unable to work, and alternative ports will be used for data transmission.

CONCLUSION

In this paper group adaptive routing in SpaceWire, GigaSpacWire andSpaceFibre networks were considered. We evaluated objectives and requirements of additional mechanisms for their realization in these standards for four case studies:

- Using of group adaptive routing for high intensity data flow transmission (between one source and one destination) is not effective due to possibility of unsuccessful selection the output port (decreasing of useful bandwidth), due to loss of big amount of data in case of disconnection in one of links, and due to possible packet reordering.

- Using of the group adaptive routing for transmission of several data flows between different independent sources and destinations via two routers (for example, which are suitated on boards of network regions) when all ports of these routers works on same speed. It showed to be efficient.

- Using of group adaptive routing for bypassing overloaded network paths is not effective if pathes between sources and destinations includes more than one router.

- Using of group adaptive routing for bypassing failed links and routers is possible when lose of packets and packets reordering is allowable. But alternative pathes should be determined for every router.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation according to the base part of the state funding assignment in 2016, project № 1810.

REFERENCES

[1] ECSS-E-50-12C. SpaceWire - Links, nodes, routers and networks. - European Cooperation for Space Standardization (ECSS), 31 July 2008

[2] ECSS-E-50-12C Rev.1 DIR3. SpaceWire - Links, nodes, routers and networks. 30 January 2016/

[3] SpaceWire Router. International SpaceWire Seminar (ISWS 2003). Steve Parkes, Chris McClements, Gerald Kempf, Stephan Fischer, Agustin Leon. 4-5 November 2003, ESTEC Noordwijk, The Netherlands.

[4] Steve Parkes. SpaceWire User's Guide. ISBN: 978-0-9573408-0-0. Published by STAR-Dundee Limited, 2012.

[5] Steve Parkes. SpaceWire for Adaptive Systems. Adaptive Hardware and Systems, 2008. AHS '08. NASA/ESA. 22-25 June 2008.

[6] FDIR Techniques for Payload Streaming Applications using SpaceWire-based Networks. Felix Siegle, Tanya Vladimirova, Jørgen Ilstad, Omar Emam. SpaceWire 2014.

[7] F. Siegle, T. Vladimirova, O. Emar, and J. Ilstad, "Adaptive FDIR Framework for Payload Data Processing Systems using Reconfigurable FPGAs," in NASA/ESA Conference on Adaptive Hardware and Systems, 2013.

[8] B. Osterloh, H. Michalik, B. Fiethe, and K. Kotarowski, "SoCWire: A Network-on-Chip Approach for Reconfigurable System-on-Chip Designs in Space Applications," in NASA/ESA Conference on Adaptive Hardware and Systems, 2008, pp. 51 – 56.

[9] INTEGRATED ONBOARD NETWORKING FOR IMA2G. Yuriy Sheynin, Elena Suvorova, Valentin Bukov, Vladimir Shurman. ICAS 2014.

[10] S. M. Parkes and P. Armbruster, "SpaceWire: a spacecraft onboard network for real-time communications." Proceedings of the 14th IEEE-NPSS conference on Real time, 2005.

[11] Evgeny Yablokov, Yuriy Sheynin, Elena Suvorova, Alexander Stepanov, Tatiana Solokhina, Yaroslav Petrichcovitch, Alexander Glushkov, Ilia Alekseev, "GigaSpaceWire – Gigabit Links for SpaceWire Networks", SpaceWire-2013. Proceedings of the 5th International SpaceWire Conference, Gothenburg 2013. Editors Steve Parkes and Carole Carrie. ISBN 978-0-9557196-4-6, Space Technology Centre, University of Dundee, Dundee, 2013, pp. 28-34.

[12] S. Parkes, A. Ferrer, A. Gonzalez, C. McClements. SpaceFibre Specification. Draft H4, February 2016.

[13] ISO/IEC JTC 1/SC 29 (2009-10-30). "Programme of Work — Allocated to SC 29/WG 11, MPEG-1 (Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s)".

[14] H.264/MPEG-4 Part 10 Tutorials (Richardson).

SpaceWire-D Prototype and Demonstration System

Networks & Protocols, Long Paper

David Gibson, Steve Parkes
Space Technology Centre
University of Dundee
Dundee, UK
d.z.gibson@dundee.ac.uk

Chris McClements, Stuart Mills
STAR-Dundee Ltd
Dundee, UK

***Abstract*—SpaceWire-D is an extension to the SpaceWire protocol that provides deterministic capabilities over existing SpaceWire equipment. The network is divided into segments using a virtual bus abstraction, where a virtual bus consists of a single RMAP initiator, one or more RMAP targets and the SpaceWire links that make up the paths between the initiator and the targets. Time-codes are broadcast periodically to provide time-division multiplexing, and a network schedule is defined by the allocation of virtual buses to time-slots. If a virtual bus has been allocated a time-slot, it is allowed to execute transactions to any of the targets within the virtual bus as long as the transactions complete their execution before the end of the time-slot. If the schedule is designed so that no virtual buses sharing a link are allocated the same time-slot, packets are no longer affected by blocking which allows the transaction execution times to be calculated and real-time constraints to be satisfied.**

The SpaceWire-D demonstration system has been designed to facilitate the verification of the draft standard. It consists of two RMAP initiators, twelve RMAP targets, a network manager device, a host PC and a routed SpaceWire network to connect the devices together. The LEON2-FT based initiator boards each contain an embedded SpaceWire-D software layer and an automated test scripting system, built on top of the RTEMS real-time operating system. The target boards respond to RMAP commands and provide event notification functionality on the backplane to allow for network activity monitoring. The network manager receives statistics and error information at the end of each schedule epoch, reported by the initiators, and informs the host PC so that it can be read, parsed and displayed to the user. Finally, the host PC runs a suite of software programs to configure, control and monitor the other devices in the demonstration system.

This paper provides an overview of the SpaceWire-D protocol and describes the design and features of the SpaceWire-D demonstration system.

***Index Terms*— SpaceWire, SpaceWire-D, Deterministic Networks, Demonstration System**

I. Introduction

SpaceWire is a data-handling network used on-board spacecraft to provide communication between scientific instruments, mass-memory storage devices, on-board computers, downlink telemetry and other subsystems [1]. SpaceWire enabled devices are connected by full-duplex data links, providing bi-directional data-flow at variable transmission rates of between 2 Mbit/s and 200 Mbit/s. The simplest SpaceWire network can consist of two nodes with a point-to-point link between them. If more complex network topologies are required, routing switches can be used to direct traffic between nodes.

SpaceWire networks can suffer from blocking caused by wormhole routing if a packet is delayed because of another packet currently using one of the links in the packet's path from its source to its destination. The packet will be held within one or more router's buffers until the links are freed and the packet can complete its journey through the network. Due to SpaceWire's arbitrary length packets, this may cause unpredictable packet propagation times which means that a regular SpaceWire network is not suitable for real-time applications such as command and control traffic because these delays could cause a critical deadline to be violated.

The aim of SpaceWire-D is to solve this problem by providing deterministic features in order to ensure that blocking does not cause deadlines to be missed, as well as allowing deterministic and non-deterministic traffic to share the same network. If these goals can be achieved, then cable mass will be reduced as the spacecraft now only requires one network to handle both payload data and control traffic which in turn will reduce complexity and cost.

II. SpaceWire-D

SpaceWire-D is a deterministic extension to SpaceWire designed by the Space Technology Centre at the University of Dundee for ESA [2].

SpaceWire-D operates by controlling which parts of the network are allowed to operate at specific times. Network time is divided into isochronous time-slots which are controlled by the distribution of consecutive SpaceWire time-codes. The network is divided into segments called virtual buses where all traffic, encapsulated within Remote Memory Access Protocol (RMAP) [3] transactions, is controlled by a single initiator. Each initiator has a schedule which describes which time-slots are allocated to its virtual buses. If some rules are adhered to when creating the schedules, the possibility of blocking can be removed and the deterministic requirements of a command and control network can be satisfied.

A. Time-Slots

A SpaceWire-D time-slot is a period of time that begins when an initiator receives a time-code and ends when the initiator receives the next time-code. SpaceWire time-codes contain a 6-bit time-value so there are 64 time-slots. This is illustrated in Figure 1.

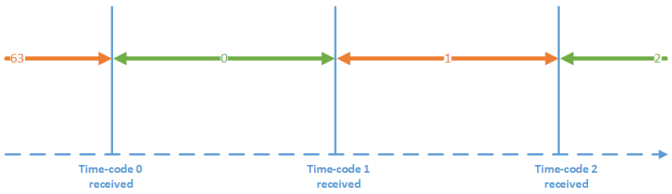


Figure 1: Time-Slots

In Figure 1, there is a timeline going left to right on the horizontal axis showing when time-codes are received by an initiator. At the start of the illustration, time-slot 63 is currently active. When time-code 0 is received by the initiator, this terminates time-slot 63 and signals the beginning of time-slot 0. The same process is repeated for another two time-codes.

The generation of time-codes is synchronised by using a single time-code master responsible for sending out time-codes at fixed-length intervals, typically at a rate of 1-1024 Hz, allowing for between 1 and 16 schedule epochs per second. Each initiator listens for time-codes being received by, for example, installing an interrupt service routine (ISR) that is called whenever a time-code interrupt is raised, or polling a time-code status flag if interrupts are discouraged. The initiator can then inform its SpaceWire-D layer that a new time-slot should be executed, which will in turn execute any scheduled transactions for the virtual bus allocated to the time-slot.

B. Virtual Buses

Virtual buses are segments of the overall network that have a specific structure. They consist of a single RMAP initiator, one or more RMAP targets and the SpaceWire links that make up the paths between the initiator and the targets. For example, take the network architecture illustrated in Figure 2.

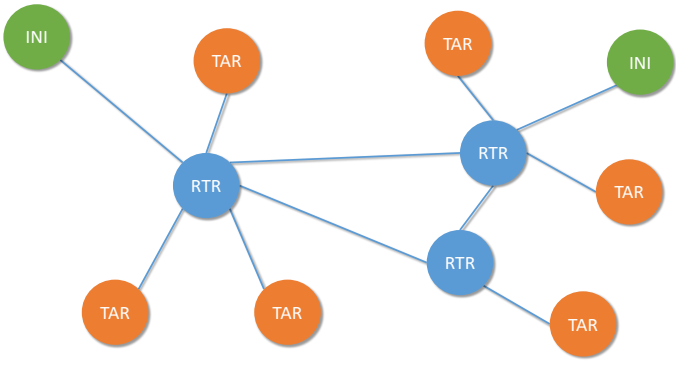


Figure 2: Overall Network Architecture

In Figure 2, there is a network containing two initiators, six targets, three routers and some links to connect the different nodes and routers. Two possible virtual bus configurations are shown in Figure 3.

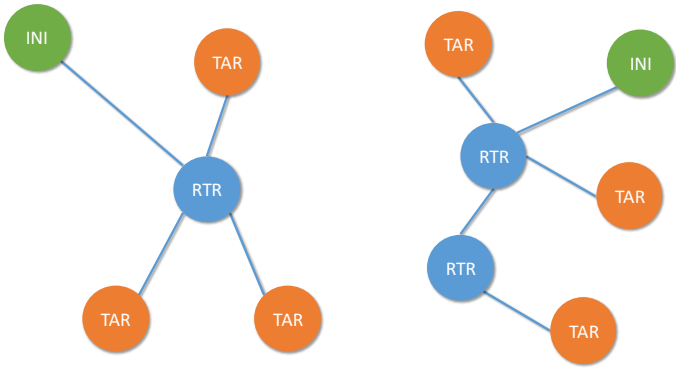


Figure 3: Example Virtual Buses

As shown in Figure 3, there are two virtual buses, each consisting of one initiator, three targets and the links between the nodes. In this example, the two virtual buses have no shared links so they can be thought of as independent i.e. they can operate at the same time without RMAP transactions on one virtual bus interfering with transactions on the other.

Virtual buses have four different functions: an initiator opens a bus, defining its configuration and allocating its time-slots; loads it with transactions, transaction groups or packet transfer requests; executes it during an allocated time-slot; and closes it when it's no longer required. There are four different types of virtual bus, each with their own implementations of the load and execute functions which provide features related to different classes of traffic which exist on a data-handling or command and control network.

1) Static Bus

The Static Bus is the simplest type of virtual bus. It is allocated a single time-slot in which it executes a repeating or single-shot transaction group.

2) Dynamic Bus

The Dynamic Bus can be allocated multiple time-slots and loaded with transaction groups. When it is loaded with a transaction group, the group is executed within the next allocated time-slot that occurs.

3) Asynchronous Bus

The Asynchronous Bus can be allocated multiple time-slots and loaded with prioritised transactions. These transactions are held in a queue and in the time-slot preceding one of the allocated time-slots, a subset of transactions is pulled from the head of the queue until no more will fit in the time-slot or the queue is empty. This transaction group is then executed in the allocated time-slot.

4) Packet Bus

The Packet Bus can be allocated multiple time-slots and loaded with requests to transfer a packet between the initiator and a target. The packet transfer operation takes place in three stages: firstly, the initiator checks the status of a packet channel within the target to make sure the target is ready to receive or send a packet; secondly, the packet is transferred in one or more segments via RMAP read or write transactions depending on if

the initiator is receiving or sending a packet; lastly, the initiator executes an EOP transaction with the target to inform it that the packet has been transferred and that the packet channel may be used to transfer another packet.

III. Demonstration System

The SpaceWire-D demonstration system consists of two LEON2-FT based PXI processor boards, acting as the initiators and controlling the execution of all RMAP transactions; three STAR-Dundee PXI RMAP interface boards [4], each containing four individual RMAP targets with separate memory regions, resulting in a total of 12 RMAP targets; one STAR-Dundee PXI RMAP interface board acting as the network manager, used to receive and store statistics and error information reported by the initiators; two STAR-Dundee PXI 8-port SpaceWire routers, providing the network connecting the devices; and one PXI system controller, running Windows 7, acting as the host PC and running a suite of software used to configure, control and monitor the other devices on the network. A photograph of the SpaceWire-D demonstration system is shown in Figure 4.

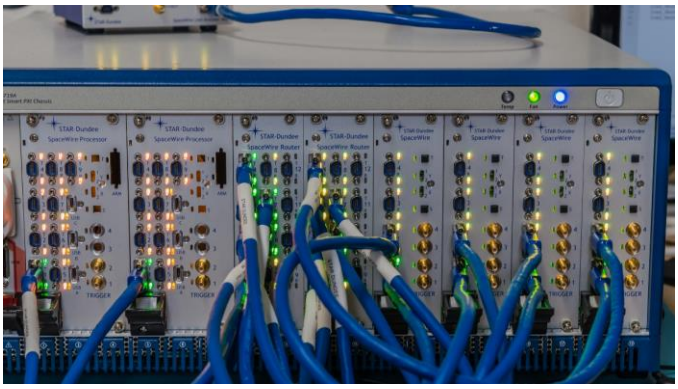


Figure 4: SpaceWire-D Demonstration System

In Figure 4, the PXI rack contains the following boards, from left to right: initiator 0, initiator 1, router 0, router 1, the network manager, target interface 0, target interface 1 and target interface 2. To the left of initiator 0, partially in shot, is the host PC.

There are 11 SpaceWire 0.5m cables providing the network between the initiators, targets, routers and network manager. The network architecture and logical addressing has been designed so that both initiators can communicate with targets on the same target interface board without sharing links. This allows, for example, Initiator 0 to communicate with two targets in Target Interface 0 and Initiator 1 to communicate with the other two targets within the same time-slot, without violating the rules of SpaceWire-D. A network architecture diagram for the SpaceWire-D demonstration system is shown in Figure 5.

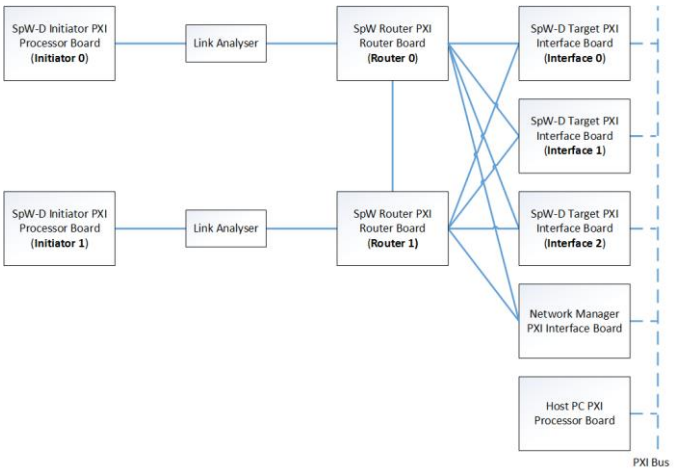


Figure 5: Network Architecture

In Figure 5, the network architecture diagram shows that initiator 0 is connected to router 0 and initiator 1 is connected to router 1. If initiator 0 wants to send an RMAP command to a target, the command is routed from router 0 to SpaceWire port 1 of the relevant target interface board and if initiator 1 wants to do the same, the command is routed from router 1 to SpaceWire port 2 of the target interface board. Commands sent to the network manager from the initiators are routed in a similar manner.

Each of the target interface boards contains four individual RMAP targets with their own logical address and region of memory. Targets 0-3, 4-7 and 8-11 are contained within interface 0, interface 1 and interface 2, respectively. The network manager uses two of its targets; the first is allocated to receive initiator 0's statistics and error reports and the second is allocated to receive reports from initiator 1.

The SpaceWire-D demonstration system uses logical addressing throughout the network to route packets between nodes. The logical addresses and the available memory regions of each device in the network are listed in Table 1.

Table 1: Logical Addresses and Memory Regions

Device	LA	Memory (Start)	Memory (End)
Initiator 0 (I)	0x30	N/A	N/A
Initiator 0 (T)	0x90	0x60000000	0x61000000
Initiator 1 (I)	0x31	N/A	N/A
Initiator 1 (T)	0x91	0x60000000	0x61000000
Target 0	0x40	0x00000000	0x10000000
Target 1	0x41	0x00000000	0x10000000
Target 2	0x42	0x00000000	0x10000000
Target 3	0x43	0x00000000	0x10000000
Target 4	0x50	0x00000000	0x10000000
Target 5	0x51	0x00000000	0x10000000
Target 6	0x52	0x00000000	0x10000000
Target 7	0x53	0x00000000	0x10000000
Target 8	0x60	0x00000000	0x10000000
Target 9	0x61	0x00000000	0x10000000

Target 10	0x62	0x00000000	0x10000000
Target 11	0x63	0x00000000	0x10000000

As listed in Table 1, each node has a logical address and, if the node is a target, a memory region. Each initiator device also contains an RMAP target with a 16 Mbyte region of memory starting at address 0x60000000 and each of the targets within the target interface boards has a 256 Mbyte region of memory starting at address 0x00000000. The target within the initiator devices is used to contain the transaction read and write buffers and to allow the host PC to write data to them before executing a test.

Figure 5 shows that the target interface boards, the network manager and the host PC are connected to the backplane PXI bus. The backplane is used by the host PC to read and write to target memory and receive RMAP command notifications from the targets, as described in Section E.

The interactions between the different devices are illustrated in Figure 6.

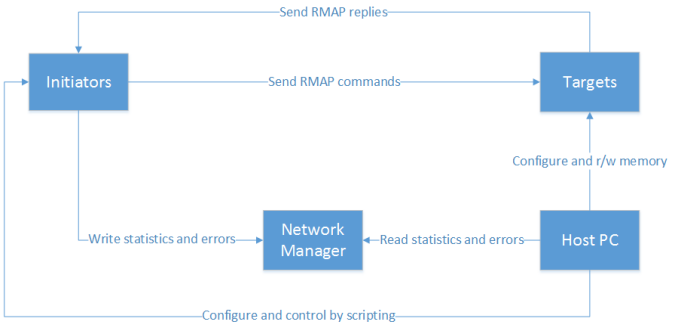


Figure 6: Device Interactions

As shown in Figure 6, each device interacts with one or more other devices in the SpaceWire-D demonstration system. The initiators send RMAP commands to the targets and the targets send RMAP replies back. The initiators report statistics and error information to the network manager, which is then read by the host PC. The host PC configures the initiators using RMAP commands and uploads automated test scripts to control their operation. The targets are configured by the host PC using a combination of RMAP commands and reading/writing to memory on the backplane.

A. Initiators

The initiators are LEON2-FT based PXI processor boards with extensive SpaceWire support. The boards have a SpaceWire router with eight external ports and three internal ports, each connected to independent SpaceWire protocol engines containing three DMA controllers, an RMAP initiator and an RMAP target.

In addition to the embedded SpaceWire-D software layer running on the initiators, which is built on top of the RTEMS real-time operating system [5], there is a demonstrator application. The application is responsible for interpreting scripted commands which are uploaded to the initiators by the Host PC in order to automate test scenarios.

The automated test scripting system allows the user to describe transactions, transaction groups, packet bus operations and time-triggered commands as a text file which is parsed, compiled and uploaded to the initiators by the host PC software. For example, an automated test script could be created that describes 10 transactions encapsulated within 2 transaction groups and a packet bus operation to send a packet from an initiator to a target. The script could then list commands to open two static buses and a packet bus at the start of the test and load them with the transaction groups and packet bus operation at specific times during the execution of the schedule. The automated test scripting system was used to implement all test scenarios during the SpaceWire-D verification activity.

B. Targets

The targets are STAR-Dundee PXI RMAP interface boards which contain a SpaceWire router with four external ports and four internal ports, each connected to an individual RMAP target. The boards have 1 Gbyte of DDR3 memory which can be divided between the four targets as configured by the user. In the case of the SpaceWire-D demonstration system, the targets are configured so that they each have access to 256 Mbytes of memory.

The target boards have the ability to notify a host application whenever certain events occur such as the execution of an RMAP command or a request for command authorisation. The notifications are sent as data structures contained within SpaceWire packets to STAR-System channel 1 on the backplane and can be received using the STAR-System API [6].

Each RMAP comment notification contains the command header parameters as well as the value of the current time-code value in the target board's router so that the time-slot in which the command was executed can be identified. In the SpaceWire-D demonstration system, this information is extracted from the SpaceWire packets by the host PC's software so that it can be used to record and display the activity between the initiators and targets as described in Section E.

C. Routers

The routers are STAR-Dundee PXI routers [4] containing eight external ports and they provide the network for the SpaceWire-D demonstration system, allowing each initiator to be routed to each interface board without sharing any links.

D. Network Manager

The network manager is another STAR-Dundee PXI RMAP interface board. It is controlled by the host PC software to act as the time-code master for the SpaceWire-D network and it also receives statistics and error information reported by the initiators via RMAP write commands to two of the targets within the board.

Each initiator is assigned a separate RMAP target and memory address to write its statistics and error information into at the end of each schedule epoch. Initiator 0 is assigned address 0x00000000 within target 0 and initiator 1 is assigned the same address within target 1.

The host PC's Network Manager software is used to listen for RMAP event notifications coming from the board, which it

parses and uses to read the statistics and error information from address 0x00000000 in the corresponding target. The information is then read from the target by the software and displayed in the Network Manager program running on the host PC. The statistics include the number of completed transactions, incomplete transactions, RMAP errors and early, late and missing time-code errors. Further error information is provided in the error list which describes the time-slot, the virtual bus related to the error and the class and type of error.

Errors are detected at three stages: firstly, if an RMAP command has incorrect header parameters or an error occurs on the initiator where the command cannot be sent, it is reported as an encoder error; secondly, if an RMAP reply is returned to the initiator with an error or if an error occurs on the initiator where the reply cannot be processed, it is reported as a decoder error; lastly, if an RMAP transaction is outstanding at the end of its allocated time-slot, it is cancelled and reported as an incomplete transaction error. The initiators are responsible for detecting and reporting the errors and the network manager is responsible for receiving the error list and informing the host PC, but no further action is taken. It is the responsibility of a higher-level protocol or the application to handle the errors.

E. Host PC

The host PC is an ADLINK PXI-3950 system controller with an Intel Core2 Duo T7500 2.2 GHz processor and 4 GBytes of 667 MHz DDR2 running Windows 7 32-bit. It is responsible for initialising the other devices within the SpaceWire-D demonstration system and running a suite of Qt4.8 based C++ applications used to configure and control the initiators, targets and network manager; and display network activity reported to the network manager via RMAP commands by the initiators, and across the backplane by the targets.

1) Initiator Configuration

The Initiator Configuration program is used to configure and control each of the LEON2-FT processor boards acting as the initiators. It has the ability to read and write the network and target parameters, used by the initiators to calculate RMAP execution times; create different types of virtual buses and assign them to the initiator's schedule; parse, compile and write automated test scripts to the initiators; and send commands to the initiators to enable and disable the schedule and other features like local-timer synchronisation.

2) Target Configuration

The Target Configuration program is used to configure and control each of the RMAP targets in the three PXI interface boards. It has the ability to read and write the RMAP command authorisation parameters; set the packet channel buffer locations and lengths; write data to, and read data from, the target memory; and enable the target interface board as a babbling node. A screenshot of the Target Configuration program is shown in Figure 7.

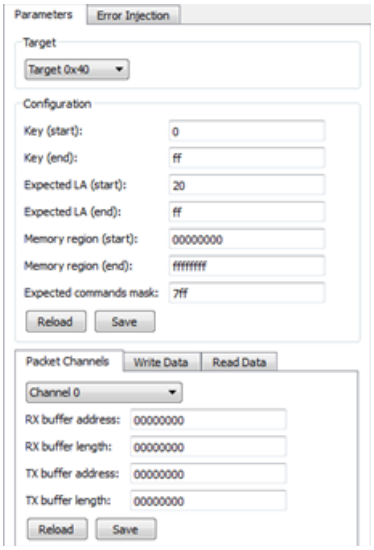


Figure 7: Target Configuration Program

In Figure 7, the top section allows the user to select which target they would like to configure. In the middle section, the authorisation parameters can be set to define the valid key range, valid target logical address range, accessible memory region and permitted commands. In the bottom section is a tab layout with three separate tabs. The first tab contains a menu to select a packet channel and fields to set the location and length of the receive and transmit buffers used by the packet bus to transfer packets between an initiator and the selected packet channel. The second and third tabs allow the user to write data to, and read data from, the target's memory. Finally, in the second main tab, the user can enable target interface boards as babbling nodes, which send out randomised RMAP commands on the network.

3) Network Manager

The Network Manager program is used to configure the time-code master and receive and display statistics and error information reported to the network manager by the initiators. It has the ability to set the time-code rate and enable or disable the time-code master; display the statistics reported by the initiator in a table, divided by type and time-slot; and display the error information as a list. A screenshot of the error list is shown in Figure 8.

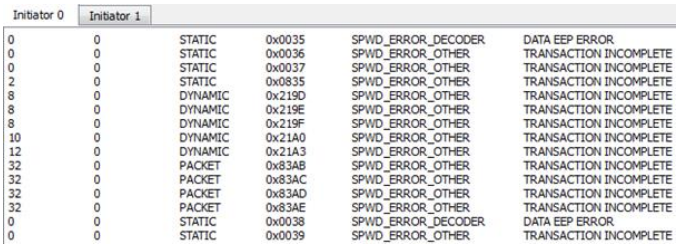


Figure 8: Network Manager Error List

In Figure 8, the screenshot shows a list of errors reported by the initiator during a test in which a STAR-Dundee Link Analyser Mk2 is periodically injecting disconnect errors in the link between router 0 and target interface 0. The columns are:

virtual bus ID, target index, virtual bus type, transaction ID, error category and error type. In this example, two types of errors are detected and reported: firstly, if the disconnect causes the command packet to be truncated, it will not have a corresponding reply so at the end of the allocated time-slot, the transaction is cancelled and reported as an incomplete transaction; and secondly, if the disconnect causes the reply packet to be truncated in its data section, it is reported as a data EEP decoder error.

When the Network Manager program is initialised, it starts to listen for RMAP event notifications from the Network Manager RMAP interface board by receiving SpaceWire packets on the backplane through STAR-System channel 1. When a notification is received, the software checks that the parameters of the RMAP command match those expected by an initiator statistics and error report. If so, the report is read from the target memory and the statistics table for the relevant initiator is updated and any errors detected during the last schedule epoch are added to the initiator's error list.

4) Target Monitor

The Target Monitor program is used to display the network activity visually and statistically through a series of views. It has the ability to display activity in real-time, by updating a grid that shows if any of the targets were read from or written to during each time-slot. It shows the number of completed transactions, bytes read from and written to the target in total and per second, and it also breaks this information up for each time-slot. Finally, it shows a list of detailed information about all RMAP transactions taking place across all targets. There are three views in the Target Monitor program: the schedule view, the target statistics view and the command list view.

A screenshot of the Target Monitor schedule view, which shows network activity as a grid, is shown in Figure 9.

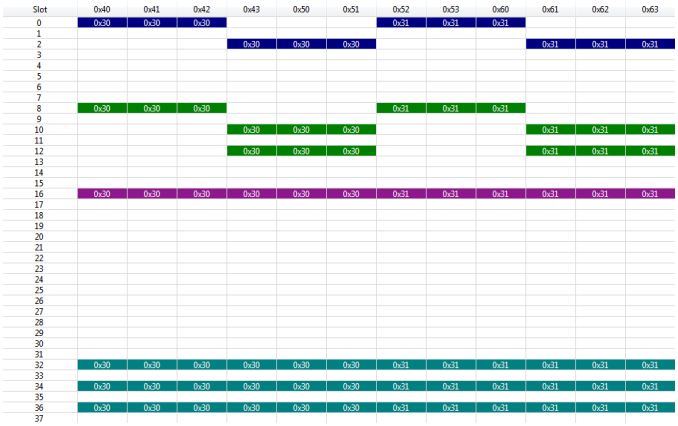


Figure 9: Target Monitor Schedule View

In Figure 9, the screenshot shows the Schedule View during the execution of a test where each initiator is executing two static buses and one dynamic, asynchronous and packet bus. The virtual buses in initiator 0 are executing transactions with targets 0x40-0x43 and 0x50-0x51, taking up the left side of the diagram. Initiator 1's virtual buses are executing transactions with targets 0x52-0x53 and 0x60-0x63, shown on the right side of the diagram. There are two static buses executed by each initiator, shown as dark blue cells, and allocated to time-slots 0 and 2. The dynamic bus executed by each initiator, shown as green cells, are allocated time-slots 8, 10 and 12. The asynchronous bus executed by each initiator, shown as magenta cells, is allocated time-slot 16. Finally, the packet bus executed by each initiator, shown as cyan cells, is allocated time-slots 32, 34 and 36.

The target statistics view lists the number of errors, commands and bytes read/written in total, per second and divided by time-slot. Figure 10 shows an image of the Target Statistics View section for target 0x40.

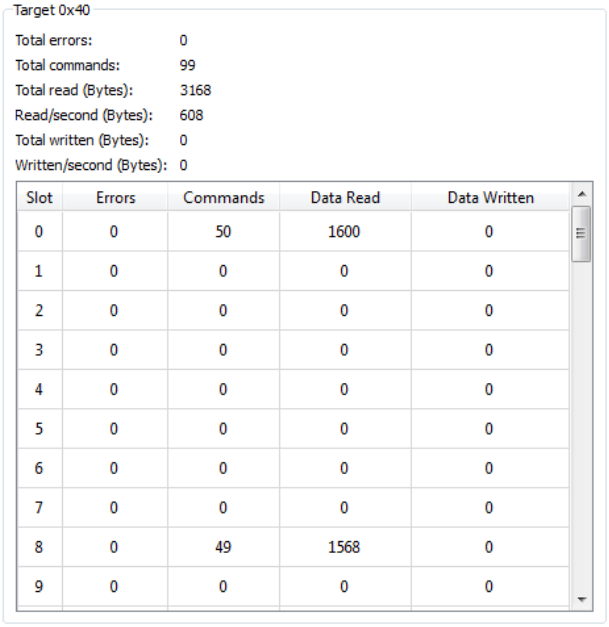


Figure 10: Target Monitor Target Statistics View

In Figure 10, the screenshot shows the Target Statistics View for target 0x40 during the execution of a schedule containing network activity in time-slots 0 and 8. The total and per second statistics are shown in the top section and the per time-slot statistics are shown in the scrollable table.

The final section of the Target Monitor program is the Command List View, which displays a detailed description of every RMAP command received on all targets. An image of the Command List View is shown in Figure 11.

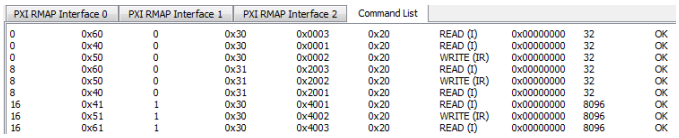


Figure 11: Target Monitor Command List View

In Figure 11, the screenshot shows the start of the Command List View during the execution of a schedule containing at least three static buses. The columns are: virtual bus ID, target logical address, target index, initiator logical address, transaction ID, RMAP key, command type, memory address, data length and status. In this case, there are nine transactions executed by static buses 0, 8 and 16. The first three, to targets

0x40, 0x50 and 0x60, and the last three, to targets 0x41, 0x51 and 0x61 are executed by initiator 0x30. The middle three, to targets 0x40, 0x50 and 0x60, are executed by static bus 8 in initiator 0x31.

IV. Conclusions

SpaceWire-D is an extension to the SpaceWire protocol that provides deterministic capabilities over existing equipment. It does this by using time-division multiplexing and a virtual bus system to schedule traffic on the network so that no blocking can occur, resulting in reliable RMAP transaction execution times.

The SpaceWire-D demonstration system was designed to verify the SpaceWire-D standard and demonstrate its capabilities. It consists of a PXI rack containing two initiators, twelve targets, a network manager, a host PC and a routed SpaceWire network to connect the devices together. An embedded SpaceWire-D layer and automated test scripting system was designed, built on top of the RTEMS real-time operating system; and a software suite, running on the host PC, was designed to configure, control and monitor the other devices on the network.

Acknowledgements

The research leading to these results has received funding from the European Space Agency under ESA contact number 4000107346/12/NL/LvH/fe. We would also like to thank David Jameux, the ESA project manager for the SpaceWire-D related activity.

References

[1] ECSS Standard ECSS-E-ST-50-12C, “SpaceWire – Links, nodes, routers and networks”, Issue 2, European Cooperation for Space Standardization, 31 July 2008, available from http://www.ecss.nl

[2] S. Parkes, D. Gibson, A. Ferrer Florit, “SpaceWire-D Standard Draft E”, Issue 0.4, Space Technology Centre, University of Dundee, April 2015

[3] ECSS Standard ECSS-E-ST-50-52C, “SpaceWire – Remote memory access protocol”, Issue 1, European Cooperation for Space Standardization, 5 February 2010, available from http://www.ecss.nl

[4] STAR-Dundee, “SpaceWire PXI”, https://www.star-dundee.com/products/spacewire-pxi

[5] The RTEMS Project, “RTEMS Real Time Operating System (RTOS)”, https://www.rtems.org/

[6] S. Mills, S. Parkes, “A Software Suite for Testing SpaceWire Devices and Networks”, DASIA International Space System Engineering Conference, Barcelona, Spain, August 2016

QoS mechanisms in SpaceFibre and RapidIO

SpaceWire networks and protocols, Long Paper

Nadezhda Matveeva, Elena Suvorova, Yuriy Sheynin
Saint-Petersburg State University of Aerospace Instrumentation
SUAI
Saint-Petersburg, Russia
nadezhda.matveeva@guap.ru, {suvorova, sheynin}@aanet.ru

Abstract—The paper presents analysis and comparison of Quality of Service (QoS) mechanisms in SpaceFibre draft H4 and in RapidIO rev 4.0. SpaceFibre and RapidIO standards are currently used in aerospace instrumentation field. SpaceFibre standard is developed by ESA, JAXA, NASA, ROSCOSMOS agencies for aerospace industry in accordance to different requirements and limitations such as a high speed data transmission, a low data transmission time, a guaranteed packet delivery, a guaranteed throughput, a possibility to establish real-time systems in conjunction with compact VLSI design and low energy consumption. RapidIO standard is developed for a high performance computing systems with a globally shared distributed memory (GSM) model. At the present time NASA applies this standard as a main standard for aerospace field.

These standards provide data transmission with different QoS. In these standards there is a feature of guaranteed packet delivery, they support several priority levels. Also they provide a mechanism of virtual channels allowing to distribute throughput between different data streams. In this paper authors have presented the information about comparison QoS mechanisms of SpaceFibre draft H4 and RapidIO rev 4.0. Moreover similarities and differences of supported technologies are demonstrated. Also author have evaluated the reachable characteristics of data transmission and overheads for each standard. Advantages and effective range of application of SpaceFibre QoS are shown.

Index Terms—SpaceWire, SpaceFibre, RapidIO, Quality of Service.

I. *INTRODUCTION*

In this paper we compare QoS mechanisms of SpaceFibre [1] and RapidIO [2] standards. We provide information about overheads for payload transmission using these standards. We analyse payload with different size. Also main features of QoS mechanisms are presented in this paper. Authors consider their advantages and disadvantages.

The rest of this paper is organized as follows: the second section presents the main features of SpaceFibre and RapidIO standards. In the third section, we show how overheads are changed according to different operation modes of the considered standards. The fourth section delivers some conclusion remarks.

II. *FEATURES OF SPACEFIBRE AND RAPIDIO STANDARDS*

In this paper, we consider of SpaceFibre and RapidIO standards. They are used in modern onboard data transmission systems.

At first, we correlate SpaceFibre and RapidIO standards to the Open Systems Interconnection model (OSI model). OSI model is a conceptual model that describes and standardizes the communication functions of a telecommunication or computing system without regard to their underlying internal structure and technology. OSI model introduces 7 layers: application, presentation, session, transport, network, data and physical. List of special functions is defined for each layer. Five layers are presented in SpaceFibre standard specification. These are Network, Data Link, Multi-Lane, Lane and Physical layers. Three global layers are presented in RapidIO standard specification. These are Logical, Transport and Physical layers. Each global layer consists of several specifications. For example, Input/Output Logical Specification, Message Passing Logical Specification, Globally Shared Memory Logical Specification are part of Logical layer. Correspondence between OSI model layers, SpaceFibre and RapidIO layers is presented on Fig.1.

Five SpaceFibre layers correspond to the three lowest OSI layers. Transport layer is not specified in SpaceFibre standard. However, it is possible to use transport layer packets of different transport protocol. For example, Remote memory access protocol (RMAP) [3] can be used. Three RapidIO global layers correspond to the four lowest OSI layers. Transport layer is specified in RapidIO standard. In this paper, we consider QoS mechanisms which are used in physical, data and network layers.

SpaceFibre QoS mechanisms are supported on Data Link layer. RapidIO QoS mechanisms are supported on Physical layer.

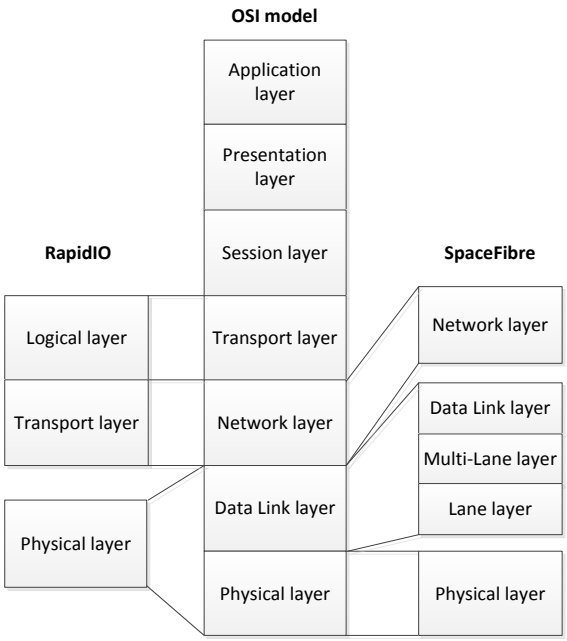


Fig. 1. *Correspondence between OSI model layers, SpaceFibre and RapidIO layers*

A. SpaceFibre QoS mechanisms

SpaceFibre provides a coherent quality of service mechanism which is able to support bandwidth reservation, scheduling, priority based qualities of service and guaranteed data delivery.

All data packets are transmitted through network using virtual channels. Maximum 32 virtual channels are supported. In particular network device implementation quantity of virtual channels can be less than 32. It is recommended to use virtual channel 0 for configuration data transmission. It must be implemented in each network device. Other virtual channels shall have a unique number in the range from 1 to 31. Each virtual channel has a priority level, reserved bandwidth and a list of time-slots during which this virtual channel can be scheduled to send data.

SpaceFibre packets can be of any size from 1 byte to infinity. The packets are split into segments before they are transferred over a data link. Each segment is sent in a data frame. Data frame consists of Start of Data Frame, data payload and End of Data Frame. Start of Data Frame and End of Data Frame have size of 4 bytes. Maximum size of data payload is 256 bytes. Data payload of frame can consist of one or several packets. Also data payload can possibly include end of one packet and start of another. Using frame mechanism with fixed maximum data payload size helps to process packets with unlimited length and at the same time provide different QoS.

Credit mechanism is used for managing the flow of information over a SpaceFibre link using one or more virtual channels with independent flow control. One flow control token (FCT) corresponds to 256 bytes. Each input virtual channel buffer contains the counter of FCT according to free buffer space. Each output virtual channel buffer shall keep track of the number of data words written into it and the number read out using an FCT credit counter, which indicates how much more data it is allowed to send. At the beginning, input virtual channel buffer sends FCTs which match its buffer space. When output virtual channel buffer receives FCT, it increments its FCT credit counter. Output virtual channel buffer can send data within current FCT credit counter. When a data segment is sent by a particular virtual channel to the medium access controller, the number of data words sent shall be subtracted from the FCT credit counter. An input virtual channel buffer shall request an FCT to be sent when the network layer reads data words from the input virtual channel buffer. Usage of this mechanism helps to avoid data words loss and retransmission of data words between output and input virtual channel buffers due to the lack of free space in input virtual channel buffer.

Priority mechanism is associated with virtual channels, not with packets. A SpaceFibre port shall support N priority levels numbered 0 to N-1, where 0 is the highest priority level. There shall be a minimum of four priority levels: 0 to 3, where priority level 0 has the highest precedence and 3 has the lowest precedence. Each virtual channel shall be able to be assigned any of the priority levels. Also it shall be possible to set more than one virtual channel to the same priority level.

Bandwidth reservation mechanism determines the precedence of a virtual channel based on the link bandwidth reserved for that virtual channel and its recent link utilization. Each virtual channel has the link bandwidth reserved for it. Virtual channel can utilize a link according to this value. The standard contains a formula which allows to determine which one of several ready virtual channels is permitted to send. Priority level and the link bandwidth reserved for the virtual channel are taken into account.

Also SpaceFibre supports scheduled quality of service. Scheduled quality of service provides a means of ensuring fully deterministic allocation of SpaceFibre network resources. Schedule mechanism is based on Time Division Multiple Access (TDMA) principles. Time is separated into a series of time-slots during which a virtual channel can be scheduled to send data. Duration of each time-slot is the same. When a time-slot arrives in which a virtual channel is scheduled, it can send data based on its precedence. During all the other time-slots, when the virtual channel is not scheduled to send data, it is not permitted to send any data even when no other virtual channel has data to send. It shall be possible for several virtual channels to be scheduled to send data in the same time-slot.

A virtual channel shall compete with other virtual channels for sending segments over the link, based on the current precedence of the virtual channel and its schedule. At first, among all virtual channels ready to send a data segment the medium access controller chooses virtual channels that have permission to send data at current time-slot. Then the precedence of a virtual channel shall be determined by its quality of service parameters such as priority level and the link bandwidth reserved for the virtual channel.

SpaceFibre supports error detection and retransmission to protect packets against loss or corruption due to transmission

errors. This mechanism provides guaranteed data transmission. All SpaceFibre data types are subject to retransmission. It is not possible to disable the retry mode. If an error in a frame is detected, then all frames in all virtual channels of port are retransmitted.

B. RapidIO QoS mechanisms

RapidIO provides the following quality of service mechanisms: priority, bandwidth reservation and guaranteed data delivery.

Data payload is transmitted in packets. RapidIO supports virtual channel technology. The protocol supports up to nine virtual channels (VC0-VC8). Virtual Channel 0 (VC0) is always active. It provides backward compatibility with previous versions of RapidIO specifications. VC0 shall be supported by all LP-Serial ports. VCs 1-8 are optional, and if implemented, may be disabled for backward compatibility. The number of optional virtual channels for VCs 1-8 may be 0, 1 (VC1), 2 (VC1, VC5), 4 (VC1, VC3, VC5, VC7) and 8 (VC1-VC8).

The LP-Serial protocol defines two methods or modes of flow control. These are named receiver-controlled flow control and transmitter-controlled flow control. Every RapidIO LP-Serial port shall support receiver-controlled flow control. LP-Serial ports may optionally support transmitter-controlled flow control.

In the receiver-controlled flow control the receiving port provides no information to its link partner about the amount of buffer space it has available for packet reception. If there is enough buffer space available, the port accepts the packet and transmits a packet-accepted control symbol to its link partner that contains the ackID of the accepted packet in its packet_ackID field. The port optionally acknowledges multiple packets with a single packet-accepted control symbol. Transmission of a packet-accepted control symbol informs the port's link partner that the packet (or packets) has been received without detected errors and that it has been accepted by the port. On receiving the packet-accepted control symbol, the link partner discards its copy of the accepted packet (or packets) freeing buffer space in the partner, [2]. In this case transmitter will repeat packet transmission when buffers space is not enough. This situation is not possible in SpaceFibre.

In transmitter-controlled flow control, the receiving port provides information to its link partner about the amount of buffer space it has available for packet reception. The value of the amount of buffer space is the number of maximum length packet buffers currently available for packet reception up to the limit that can be reported in the field, [2]. If transmitter get status information from receiver during packet transmission, then transmitter get not accurate information about free receiver buffer space. Packets transmitted in excess of the free_buffer_count are transmitted on a speculative basis and are subject to retry by the transmitter. That results in a number of retries and discarded packets can reduce the effective bandwidth of the link. Also such flow control approach can lead to ineffective buffer space utilization for series of short packets. In comparison with RapidIO, SpaceFibre's credit counter is measured in bytes. It helps to control available receiver buffer space precisely and avoid data retransmission due to the lack of buffer space in a link receiver.

RapidIO supports packet transmission based on priority. Only packets within VC0 have ordering rules based on priority, all other virtual channels do not support them. Maximum number of priorities is 8. Packet priority is formed on LP-Serial Physical Layer Specification. It is based on the flow identifier set on Logical Layer. There are tables for mapping flow identifier (flowID) of the transaction into the priority field (and optionally the CRF bit) of the packet. Also it is interesting to notice, that transaction requests that require responses, and their corresponding responses, must use VC0 with the appropriate priority.

Bandwidth reservation mechanism is partially provided in RapidIO. In comparison with SpaceFibre, rules for control bandwidth reservation in RapidIO are not specified by the specification. These rules are vendor dependent. According to Part 6: LP-Serial Physical Layer Specification 3.1 section 6.11 "Transaction and Packet Delivery Ordering Rules" the whole link bandwidth is uniformly divided into 'N' portions and each portion is 1/N of the whole link bandwidth. Each VC is configured to have a guaranteed bandwidth. The method of bandwidth division among VCs is also vendor dependent. But VC0 may be treated with strict priority, getting whatever bandwidth is required when it has traffic to transport. In this condition, the remaining VCs will divide up whatever portion of bandwidth remains, [2].

As well as SpaceFibre, RapidIO supports error detection and retransmission to protect packets against loss or corruption due to transmission errors. In comparison with SpaceFibre, RapidIO provides unreliable delivery of packets. This means that packets are not retransmitted and virtual channel operates in continuous traffic (CT) mode. VC0 should always operate in reliable traffic (RT) mode. Any of VC1 through VC8 that are implemented shall support operation in RT mode and may optionally support and be configured for operation in CT mode, [2]. RT virtual channels operate as a "RT Group". It means that when the error recovery protocol is used to recover a damaged packet, the unacknowledged packets for all virtual channels in RT mode are retransmitted.

III. QoS ESTIMATION FOR SPACEFIBRE AND RAPIDIO STANDARDS

A. Overhead estimation for data transmission

In this section, we present overhead estimation for data transmission using SpaceFibre and RapidIO standards. At first we determine data transmission parameters for these standards. RapidIO links operate at Baud Rate Class 1. Baud Rate Class 1 is used for lanes running at 1.25 Gbaud, 2.5 Gbaud, 3.125 Gbaud or 5 Gbaud (1, 2, 2.5, 4 Gbps). 8b/10b encoding scheme is used for Baud Rate Class 1. Also 8b/10b encoding scheme is used in SpaceFibre. The overheads occurring due to this encoding scheme are not considered in this paper. Control Symbol 24 is used in RapidIO for Baud Rate Class 1. This means that each control symbol has size of 24 bit. Packet Delimiter Control Symbol (/PD/) or Start of Control Symbol (/SC/) are used to delimit a control symbol. They both have the size of 1 byte.

Control symbol with additional delimiter has the size of 4 bytes. Control symbol of SpaceFibre has the size of 4 bytes as well.

We use the following format of packet to compare data transmission overheads in SpaceFibre and RapidIO standards. Type 6 (SWRITE transaction) is RapidIO packet (Fig. 2). Write command (Fig. 3) formed by RMAP transport protocol is used in SpaceFibre.

We use data payloads of different size in our research. Size of data payload constitute 16, 32, 64, 256 bytes, 2KB, 20 KB, 1 MB and 2 MB. We analyze two cases. In the first case, we estimate minimum overhead when all packet's fields have minimum size and the packet does not contain optional fields. In the second case, we estimate maximum overhead when all packet's fields have maximum size and the packet includes all possible optional fields. We analyze data transmission in single lane mode.

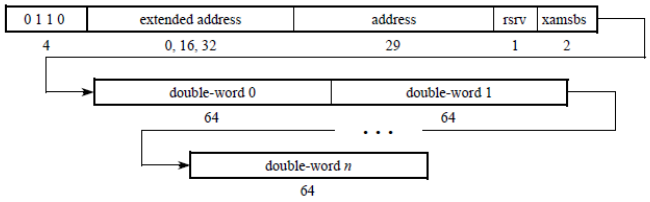


Fig. 2. *Format of RapidIO packet*

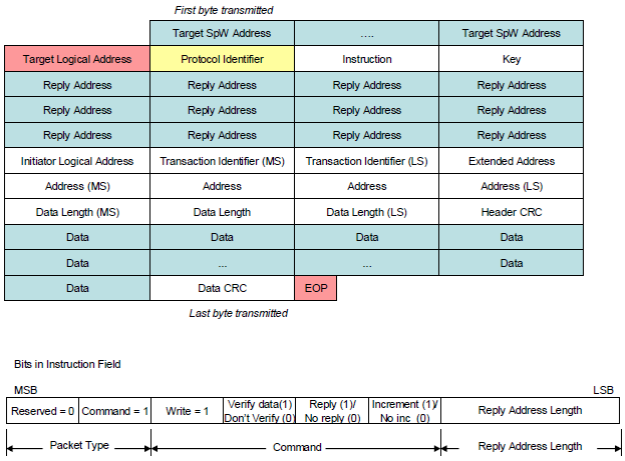


Fig. 3. *Format of RMAP write command*

On the Fig. 4 and Fig. 5 the relation between overheads and payload is shown for short (16-64 bytes) data payloads when overheads have minimum and maximum possible values correspondingly. From these figures, it can be concluded that for minimum and maximum overheads the relation between overheads and payload for RapidIO is less than for SpaceFibre when operating in the mode of remote memory write. For short packets (16-64 bytes of data payload) and lowest possible overheads for RapidIO the overheads constitute from 162% to 40% of the payload, while for SpaceFibre - from 225% to 56% respectively. For short packets (16 to 64 bytes of data payload) and maximum possible overheads for RapidIO they constitute from 225% to 56% of the payload, while for SpaceFibre – from 375% to 93%. In case of bigger payload, the relation decreases as expected.

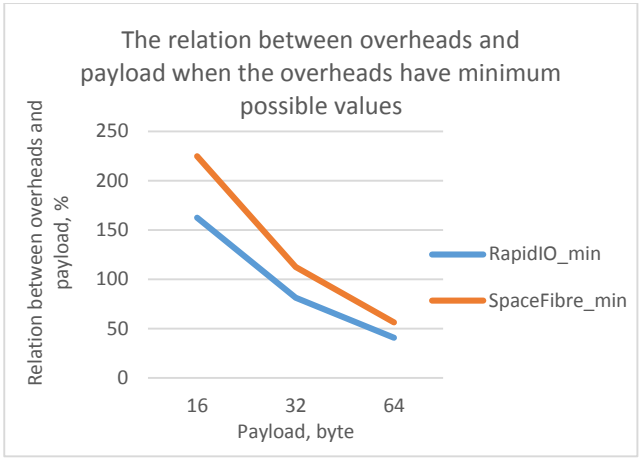


Fig. 4. *The relation between overheads and payload when the overheads have minimum possible values*

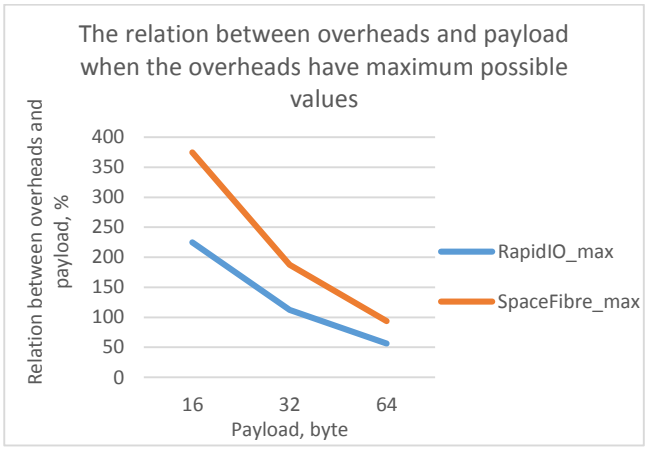


Fig. 5. *The relation between overheads and payload when the overheads have maximum possible values*

On the Fig. 6 the overheads for medium (256 bytes, 2KB and 20KB) size data payloads are shown for the cases when the overheads have minimum and maximum possible values correspondingly. In accordance to SpaceFibre standard, considered data payload is transmitted in one RMAP packet. This packet is divided into frames on Data link for further data transmission process. Size of each frame payload is 256 bytes. Also start of frame and end of frame are added. Size of start of frame and end of frame is 4 bytes accordingly. In accordance to RapidIO standard, considered data payload is divided into several packets with maximum size of payload (256 bytes). Each packet has its own header. From Fig. 6 it can be concluded that for minimum and maximum overheads the overheads for SpaceFibre is less than for RapidIO in case when data payload is 20 KB. On the Fig. 7 and Fig. 8 the relation between overheads and payload is shown for medium (256 bytes, 2KB and 20KB) size data payloads when the overheads have minimum and maximum possible values correspondingly.

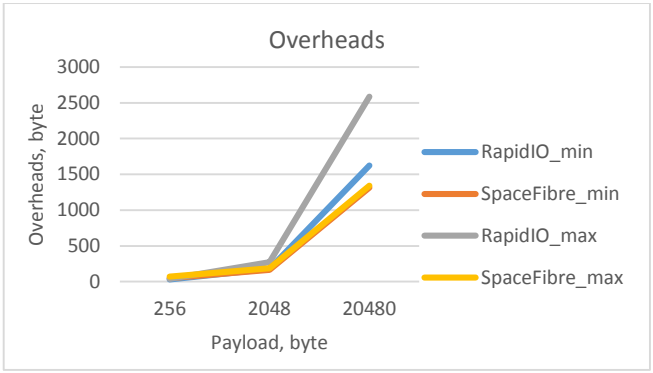


Fig. 6. *Overheads for medium size data payload*

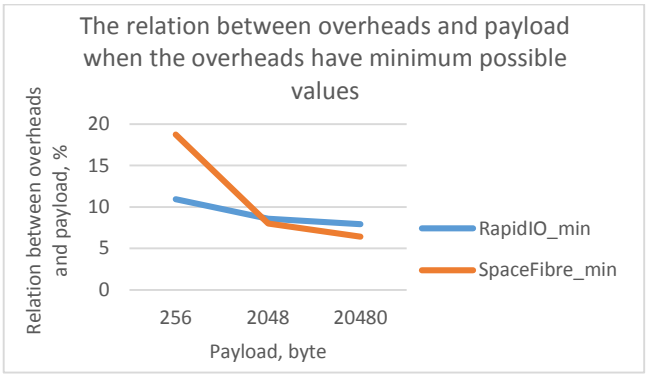


Fig. 7. *The relation between overheads and payload when the overheads have minimum possible values*

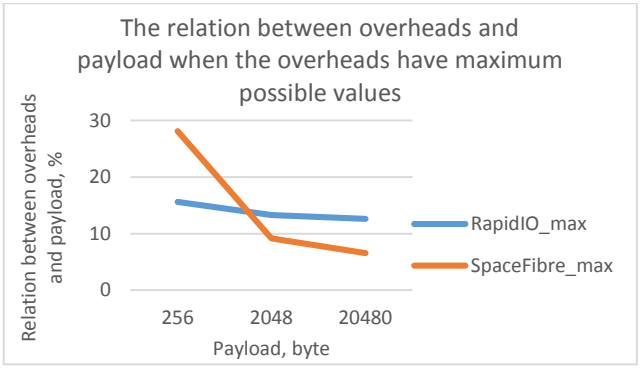


Fig. 8. *The relation between overheads and payload when the overheads have maximum possible values*

On the Fig. 9 the overheads for large (1MB, 2MB) size data payloads are shown for the cases when the overheads have minimum and maximum possible values correspondingly. In accordance to SpaceFibre standard, considered data payload is transmitted in one RMAP packet. This packet is divided into frames on Data link for further data transmission process with additional overheads as in the previous case. In accordance to RapidIO standard, considered data payload is divided into several packets as in the previous case. From Fig. 9 it can be concluded that for minimum and maximum overheads the overheads for SpaceFibre is less than for RapidIO in case when data payload is large. On the Fig. 10 and Fig. 11 the relation between overheads and payload is shown for large (1 MB, 2MB) size data payloads when the overheads have minimum and maximum possible values correspondingly.

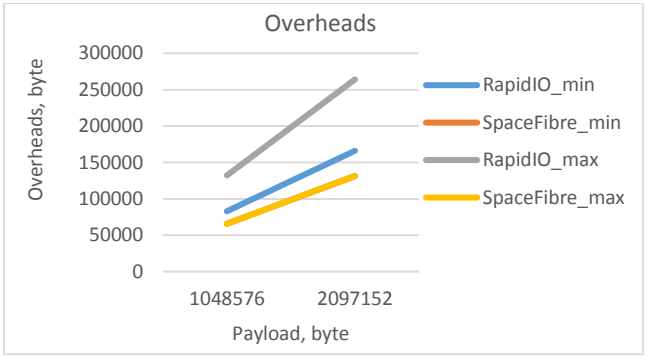


Fig. 9. *Overheads for large data payload*

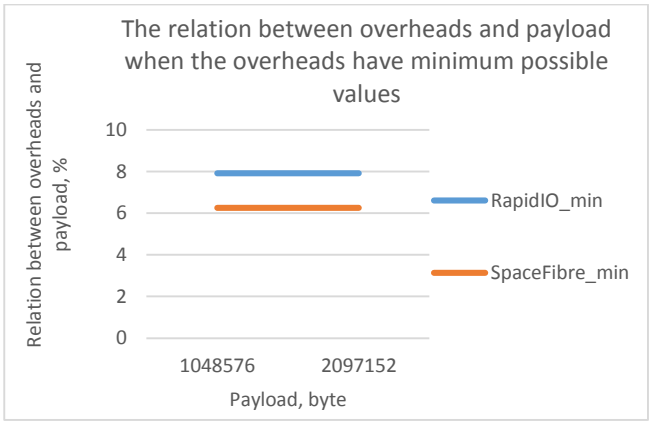


Fig. 10. *The relation between overheads and payload when the overheads have minimum possible values*

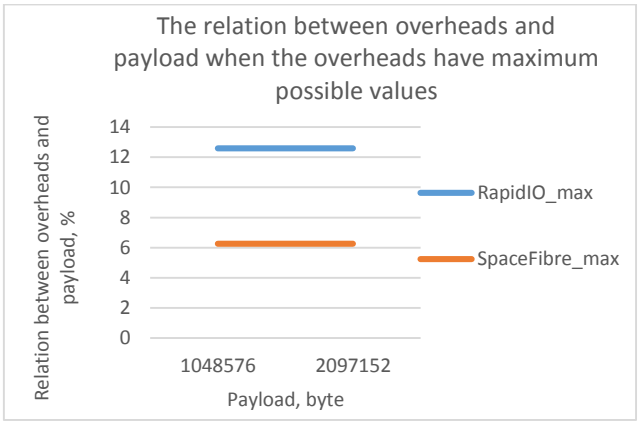


Fig. 11. *The relation between overheads and payload when the overheads have maximum possible values*

From the previous figures it can be concluded that for large size data payloads SpaceFibre standard is more efficient than RapidIO from the overheads point of view. However, for short size data payloads RapidIO is more efficient.

Besides RMAP packets, SpaceFibre supports data transmission in SpaceWire packets. Format of SpaceWire packet is presented on Fig.12. Each packet contains destination address, cargo/payload and end of packet. The destination address shall consist of a list of zero or more destination identifiers, [3]. The case of zero destination identifiers in the destination list (i.e. the destination list is empty) is intended to support a network which is simply a single point-to-point link

from source to destination. We do not consider this case. A destination identifier shall comprise one byte. The cargo shall contain one or more bytes. Maximum size of cargo is not specified by SpaceWire standard. Size of end of packet is 1 byte. Number of destination identifiers in list depends on routing rules. When network supports routing table it may be enough to have only one destination identifier. When path routing is used in the network, then number of destination identifiers depends on the length of data transmission path.

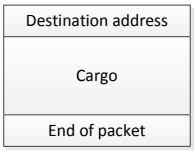


Fig. 12. *Format of SpaceWire packet*

When comparing RapidIO's and SpaceFibre's with SpaceWire packets we use SpaceWire packet which consists of destination address (1 byte), cargo/payload (from 16 bytes to 20 KB) and end of packet (1byte). Also we take into account control symbols such as ACK, FCT which are used for data transmission process in SpaceFibre standard.

Below we present figures where we show overheads for data payloads when SpaceFibre and RapidIO standards are used. From Fig. 13 it can be concluded that overheads for SpaceFibre are 1.4 times less than for RapidIO in case when data payload has size of 16,32,64 bytes. On the Fig. 14 the relation between overheads and payload is shown for short (16-64 bytes) size data payloads when overheads have minimum possible values for RapidIO and SpaceWire packet is transmitted by SpaceFibre. From this figure, it can be concluded that the relation between overheads and payload for RapidIO and SpaceFibre is the same for payload of 64 bytes.

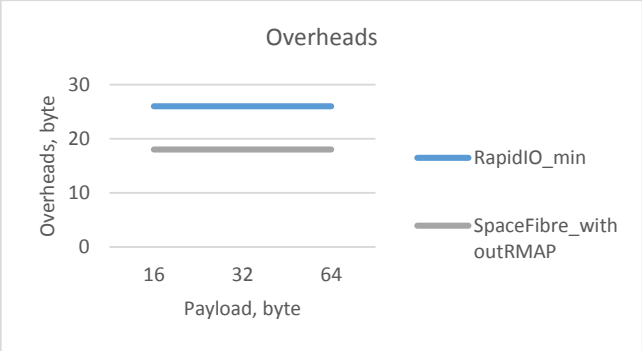


Fig. 13. *Overheads for short data payload*

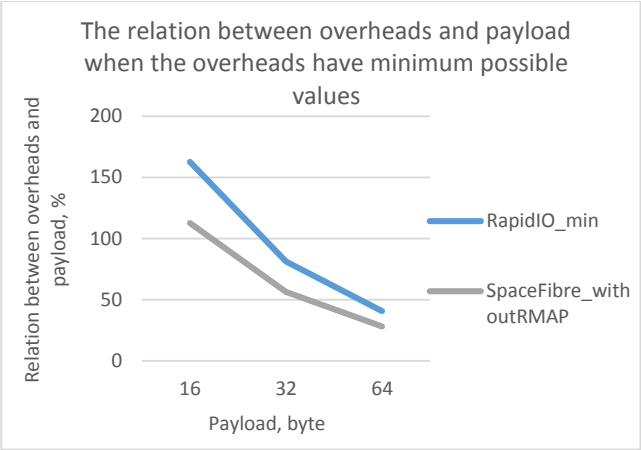


Fig. 14. *The relation between overheads and payload when the overheads have minimum possible values for short data payload*

From Fig. 15 it can be concluded that overheads for SpaceFibre and RapidIO are the same for data payload of 256 and 2048 byte sizes. But overheads for RapidIO are 1.25 times bigger than overheads for SpaceFibre in case when data payload has size of 20 KB.

On the Fig. 16 the relation between overheads and payload is shown for medium (256 bytes-20 KB) size data payloads when overheads have minimum possible values for RapidIO and SpaceWire packet which is transmitted by SpaceFibre. From this figure, it can be concluded that the relation between overheads and payload for SpaceFibre is less than for RapidIO for payload which has size of 20 KB and more.

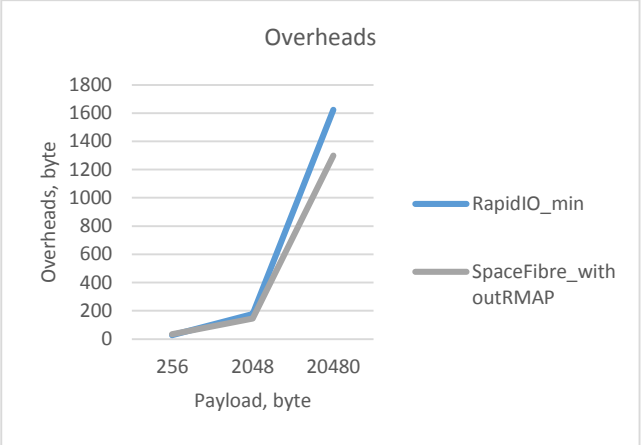


Fig. 15. *Overheads for medium data payload*

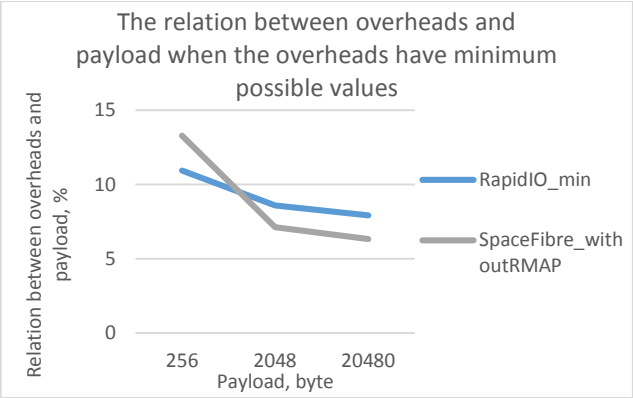


Fig. 16. *The relation between overheads and payload when the overheads have minimum possible values for medium data payload*

B. *Evaluation of the possibility of several data flows transmission in one virtual channel*

One terminal node can send several data flows in one virtual channel according to RapidIO standard due to the limited number of virtual channels. For example, different applications use one virtual channel (VC0) when it is required to get response transactions on logical layer of RapidIO. Number of virtual channels in SpaceFibre is more than in RapidIO. However, number of virtual channels in SpaceFibre systems is limited due to hardware costs. Therefore, the situation is possible when several data flows transmit using the same virtual channel with the similar QoS characteristics.

In accordance to SpaceFibre and RapidIO standards, the case is possible when different data flows being transmitted in the same virtual channel of different output ports, can be merged into one input port. Example of this situation is presented on the Fig.17. Maximum size of RapidIO packet is 256 bytes. It helps to predict the delay of packet transmission in RapidIO network.

Maximum size of SpaceFibre packet is not limited. Therefore, when we send a long packet, the delay of packet transmission can hardly be predicted for the situation when several data flows are merged into one virtual channel. For example, “orange” data flow and “blue” data flow use virtual channel 3 for data transmission. These data flows are merged in port 3. Packet 2 waits until the whole packet 1 is transmitted. However, developer of SpaceFibre network can take into account this information and can use packets of optimal size in accordance to system requirements.

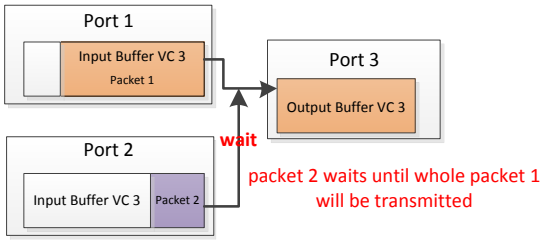


Fig. 17. *Graphical representation of single packet waiting for transmission on the background of the another packet transmission in the same virtual channel*

IV. *Conclusion*

TABLE I. Comparison SpaceFibre and RapidIO

Parameter	**SpaceFibre**	**RapidIO**
Maximum packet size	not limited	payload 256 bytes
Data flow mechanism	Credit mechanism. One flow control token (FCT) corresponds to 256 bytes. Input virtual channel buffer overflow is impossible.	Supports receiver-controlled flow control and transmitter-controlled flow control. For the receiver-controlled flow control the receiving port does not provide information to its link partner about the amount of buffer space it has available for packet reception. For transmitter-controlled flow control one credit corresponds one packet with maximum size. Input virtual channel buffer overflow is possible in both modes.
Support priority quality of service	yes	yes
Minimum number of priorities	4	1
Maximum number of priorities	not specified	16
Priorities are associated with	virtual channel	packets which are transmitted in VC0
Support of virtual channel mechanism	yes	yes
Minimum number of virtual channels	1	1
Maximum number of virtual channels	32	9
Suport bandwidth reservation	yes	yes
Standard determines rules for control bandwidth reservation	yes	no
Support guaranteed data transmission	yes	yes
Is it possible to transmit data without guaranteed data transmission	no	yes, it is optional for VC1-VC9. VC0 supports only guaranteed data transmission
Support scheduled quality of service	yes	no
Support CRC	yes (16 bit)	yes (16 bit)

SpaceFibre supports bigger variety of QoS types than RapidIO. SpaceFibre is more flexible in regard to the number of virtual channels supported in each device, their numeration, opportunities of fine tuning of QoS. For a system developer this provides additional possibilities for network’s development and configuration and therefore allows to create a network which fits precisely to the data flows being transmitted. At the same time the overheads in case of short packets transmission in SpaceFibre can appear a little bit bigger than in RapidIO with standard SpaceWire transport protocols (RMAP for example). On the other hand, it is important to note that a high speed

network usually has the biggest load when transmitting data flows consisting of long packets. In this case the useful throughput of SpaceFibre appears to be bigger.

The list of drawbacks of SpaceFibre compared to RapidIO includes mandatory retransmission in data link which cannot be disabled as well as the necessity of additional mechanism controlling packets length when several data flows are transmitted using one virtual channels (RapidIO has such mechanisms on Logical level). These aspects should be taken into account when improving SpaceFibre standard.

ACKNOWLEDGMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation according to the base part of the state funding assignment in 2016, project № 1810.

REFERENCES

[1] SpaceFibre Specification Draft H4, April 2016

[2] RapidIO™ Interconnect Specification Revision 4.0, June 2016.

[3] ECSS ‑ E ‑ ST ‑ 50 ‑ 52C SpaceWire - Remote memory access protocol, February 2010

SpaceFibre 2 (Long)

SpaceFibre Multi-lane

SpaceFibre, Long Paper

Albert Ferrer Florit, Alberto Gonzalez Villafranca
STAR-Dundee Ltd
STAR House, 166 Nethergate
Dundee, DD1 4EE, UK
albert.ferrer@star-dundee.com

Steve Parkes
Space Technology Centre, University of Dundee,
Dundee, DD1 4EE, UK
sparkes@computing.dundee.ac.uk

***Abstract*— SpaceFibre is a multi-Gbits/s, on-board network technology for spaceflight applications, which runs over electrical or fiber-optic cables. SpaceFibre supports multi-lane, thus allowing data to be sent over several individual physical lanes to enhance throughput and robustness. This is required by new generation payloads, such as SAR and multi-spectral imaging instruments. This paper describes the development of the multi-lane capabilities of SpaceFibre and its successful hardware implementation on space-qualified devices. The protocol has been designed to work with an arbitrary number of bidirectional or unidirectional lanes. In the event of a lane failing, SpaceFibre multi-lane mechanism supports hot redundancy and graceful degradation by automatically spreading traffic over the remaining working lanes. User data transfer is resumed in just a few microseconds without any data loss. These advanced capabilities are not provided in other high-speed link protocols available for space applications.**

***Index Terms* — SpaceFibre, multi-lane, hot redundancy, SpaceWire, Networking, Spacecraft Electronics.**

I. Introduction

SpaceFibre is a new technology for use onboard spacecraft that provides point-to-point and networked interconnections at Gigabit rates with Quality of Service. SpaceFibre interoperates seamlessly with a SpaceWire network over virtual channels, as it uses the same packet definition. It provides broadcast capabilities and it is able to operate over a copper or fiber-optic communication medium.

New generation payloads, such as SAR and multi-spectral imaging instruments, require the use of multiple parallel high-speed links to fulfil the increasing bandwidth requirements [1]. To accommodate these needs, SpaceFibre supports multi-lane operation, thus allowing data to be sent over several individual physical lanes to enhance throughput and robustness.

This paper describes the development of the multi-lane capabilities specified in the SpaceFibre Standard [2] and its hardware implementation on radiation hardened space-qualified FPGAs.

Multi-lane is an optional capability of a SpaceFibre link defined in the Multi-Lane layer of the SpaceFibre protocol stack. As shown in Fig. 1, the Multi-Lane layer is defined between the Data Link layer and the Lane layer implemented for each available lane.

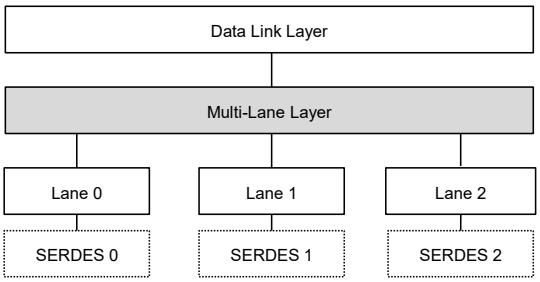


Figure 1. SpaceFibre Multi-Lane layer

The Data Link layer provides quality of service and flow control for a SpaceFibre link. It frames the information to be sent over the link to support QoS and multiple virtual channels. It also provides error recovery capabilities, detecting any frames or control words that go missing or arrive containing errors and resending them.

The Lane Layer establishes a connection across a SpaceFibre lane using a lane initialization state machine. This ensures that bit, symbol and word synchronizations are achieved and that the two ends of the lane are both ready to send and receive data with a nominal Bit Error Rate. The Lane Layer also encodes data and control words into 8B/10B symbols, sends and receives symbols over the lane and decodes the received symbols into data and control words.

The Multi-lane layer coordinates the operation of multiple lanes as a single SpaceFibre link, providing higher data throughput and redundancy. Because the logic that initialize a lane and monitor its status is located below the multilane layer, each lane can be initialized and operated independently of each other.

This architecture also supports the implementation of graceful degradation, which means that in the event of one or more lanes failing, traffic is spread over the remaining working lanes automatically. When combined with Data Link layer QoS, the bandwidth allocated to lower priority virtual channels is reduced when required to ensure that most important information gets through and deterministic traffic is maintained. Bandwidth overprovision and dynamic power management is also possible. These capabilities are very useful for space applications where strict power constrains and a high level of reliability is required on the harsh space environment.

The multi-lane requirements are expanded and consolidated in section 2. Section 3 describes the protocol analysis and the design of the SpaceFibre multi-lane capabilities. Section 4 shows the hardware implementation. Finally, conclusions are made in section 5.

II. Multi-lane requirements

The multi-lane capabilities of SpaceFibre have been designed to meet the following end-user requirements:

- Support an arbitrary number of lanes. This allows redundancy and graceful degradation without any restriction on the number of lanes.
- Re-synchronize both ends when the number of lanes changes, without resetting any lane, and as fast as possible. This way user data can be easily buffered when a lane is added or removed, until the link is again ready.
- Support hot redundancy.
- Support dynamic unidirectional lanes to save power and mass for asymmetric user data flows.
- Robust against lane errors and misconfiguration.
- Keep the same protocol overhead than single lane configuration.
- Number of lanes must be independent on the port width of the end-user interface.

These requirements enable unique capabilities for SpaceFibre. Other high speed protocols have limitations in the redundancy mechanisms. For example, in RapidIO [3] when one lane fails, the link falls back to a single lane. PCI Express [4] allows the link to continue using more than one lane, but it takes time as the link needs first to be reset. Interlaken [5] allows an arbitrary number of lanes to operate but does not define a mechanism for link reconfiguration when a lane fails, as this is expected to be done by software.

Fig. 2 shows a use case enabled by the above requirements. In this setup, unidirectional lane 6 can be enabled when one lane fails or higher data rate is required and bidirectional lane 2 can be set as a unidirectional lane for power saving reasons. Note that at least one bidirectional lane must be working for the link to operate.

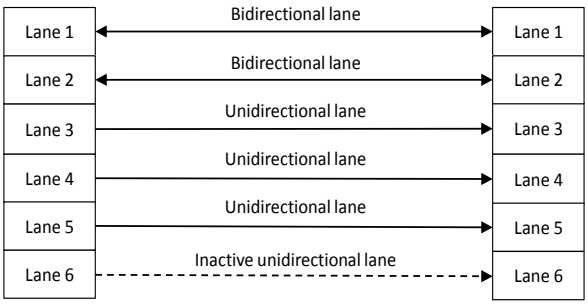


Figure 2. SpaceFibre Multi-Lane layer use case.

There are additional requirements related with SpaceFibre standardization efforts:

- Single-lane SpaceFibre implementation must not be affected by new rules added by multi-lane capabilities. This ensures that legacy single-lane implementations are still compatible with future single-lane implementations.
- Minimize modifications to the definition of other layers in the SpaceFibre standard.

Finally, there are the requirements regarding SpaceFibre implementation on space qualified devices:

- Must be feasible to implement in radiation hardened FPGAs, which are slower than state of the art COTS components.
- Resource usage in radiation hardened FPGAs must be minimized.

III. Multi-lane Design

The multi-lane specifications of SpaceFibre were designed with the following methodology:

1. Identify and evaluate the key concepts that could allow the requirements to be met (e.g. define generic protocol sequence diagrams).
2. Constrain these concepts to work with the more specific set of rules already specified in the SpaceFibre standard (e.g. adapt to control word definitions)
3. Validate the concepts and derived new set of rules in a prototype using a software simulator that can easily be modified. If an issue is found, rework the concept and/or associated set of rules.
4. Refine the simulation engine until it validates with high accuracy the proposed multilane specifications.

The resulting main concepts and specifications are explained in the following subsections.

A. Distribution of control and data words over sending lanes

In this specification, a row is defined as the set of words sent over all sending lanes simultaneously. These words can be data or control words. Data words contain data from the user interface and control words are generated by the Data Link layer to support the protocol operation.

In a single-lane SpaceFibre implementation, control words are processed at a rate of 62.5Mhz for a 2.5Gbps link rate, as there can only be one single control word for every 40 bits. If a row can contain multiple control words, then the processing rate required would increase with the number of lanes [6]. It would then not be possible to implement multi-lane in some radiation hardened devices. Therefore we must enforce that a row can only contain one control word.

The simplest solution is to replicate each control word to be sent across all lanes in a row. This avoids to have complex rules that deal with mixing data and control words in the same row. Fig. 3 shows this solution and the use of the PAD control word when the size of a data frame is not a multiple of the number of lanes available.

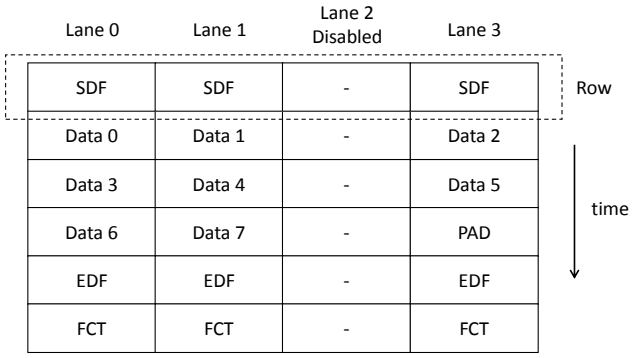


Figure 3. Words forming a row across a multi-lane link

In order to keep the same protocol overhead than single lane implementations, the maximum data frame size needs to be increased. The maximum data frame size was defined as a trade off between latency and protocol overhead. In a multi-lane solution, the maximum data frame size can be increased without modifying these metrics.

Another advantage of this solution is that the data frame CRC can be computed lane by lane as the CRC is provided in the EDF control word. A different CRC value can be included in the EDF of each lane. This keeps intact the error detection capabilities of the CRC for burst errors and for the amount of data covered by the CRC. More important, this solution simplifies the computation of the CRC in slow FPGAs as the incoming rate of the data covered by the CRC is kept the same than in single-lane implementations.

B. Lane alignment at reception

At the receiver side, the set of words received from each lane need to be aligned to compensate for small differences in lane delays. This delays are due to different cable lengths or line driver delays. Therefore, data and control words can not be passed to the Data Link layer until the multi-lane layer has compensated this skew and it is processing the same original rows sent by the sender side.

The lane alignment is usually done with a set of FIFOs that compensate the delays of each lane, using a specific control word, called ALIGN, that is known to be sent over all lanes simultaneously. To cover the case of lane errors, the ALIGN word needs to be sent periodically and with a minimum separation in between. When the alignment is no longer needed, the sending of ALIGNs can be disabled to avoid increasing the protocol overhead.

Fig. 4 shows the set of words received with a skew between lanes, which is compensated in Fig. 5 using ALIGN control words.

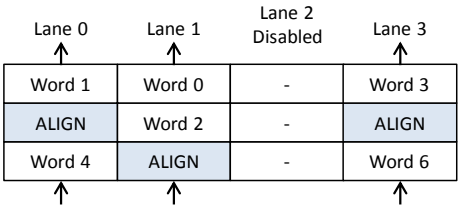


Figure 4. Rows not aligned at reception

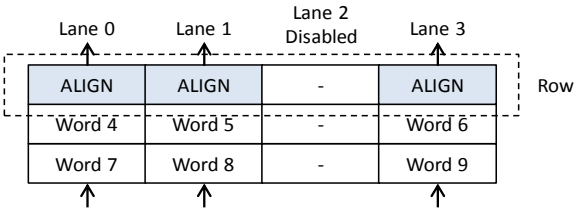


Figure 5. Aligned rows at reception

To avoid data corruption, data words in a row should be processed in the same order than they were placed in the row by the sending side. To help with this requirement, each lane has a lane number associated and it is enforced that words are processed starting with the lowest lane number. Then, this requirement can be fulfilled if the receiver side knows two parameters:

a) The lane number of each lane

b) The total number of lanes used by the sending side.

This information is provided within the ALIGN word itself, so the information is up to date when the receiver side use these words for the alignment procedure.

C. Alignment state machine

It has been stated that data and control words can not be passed to the Data Link layer until the multi-lane layer has completed the alignment process and both sides are aware of the lanes used for sending and receiving. This means that this process has to be performed each time the number of lanes in active state, i.e. working lanes, changes due to a lane error or a lane being enabled or disabled by the user.

The alignment state machine ensures that no data is being transferred to the Data Link layer when the lanes are not aligned and that ALIGN words are only sent when it is required. Three states are defined:

a) *Not Ready*: lanes have not been aligned and only ACTIVE and ALIGN words are sent. This state is set when the number of lanes in active state changes or an alignment error is detected.

b) *Near-End Ready*: lanes are aligned and the Data Link layer is being used to send and receive data and control words. However, ALIGN words are sent as the far-end has still not sent any data word.

c) *Both Ends Ready*: data words were received indicating that the far-end has aligned the lanes. No ALIGN words are sent. Link is ready.

The ACTIVE control word sent by the multilane layer has three functions:

a) Stop the flow of words from the Data-Link layer when the lanes are not aligned.

b) Indicate to the far-end that the lanes are not aligned.

c) Indicate in the cargo of this word which lanes are in active state, so the other end can synchronize the active state of the lanes. This is especially important in unidirectional transmit lanes, which can not detect if the receiving side has disconnected the lane. In a bidirectional lane, the lane initialisation state machine can detect if the other side exits the active state.

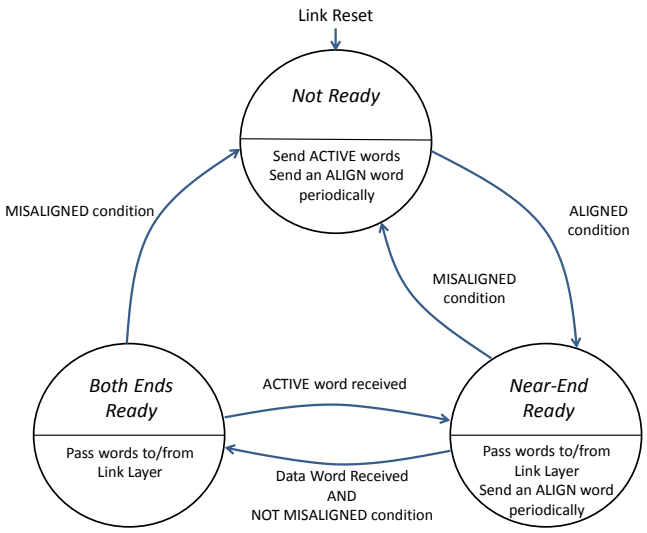


Figure 6. Alignment state machine

Fig. 6 shows the alignment state machine. The ALIGNED condition is asserted when all receiving active lanes are aligned, these lanes are the same active lanes indicated by the received ACTIVE words and they are consistent with the content of the ALIGN words. The MISALIGNED condition is asserted when the active lanes change in the near end or far end or there is an error related with the alignment process.

Fig 7 shows a protocol sequence diagram describing what occurs when one lane fails at the far end and becomes not active. The near end detects this event when it receives the ACTIVE words. It then moves to Not Ready state and starts sending ACTIVE words too. When the alignment process is completed using ALIGN words, the state machines move to Near-End Ready state. Finally when data words are received indicating the other side is in Near-End Ready, the state machines move to Both Ends Ready state and the link is considered to be ready.

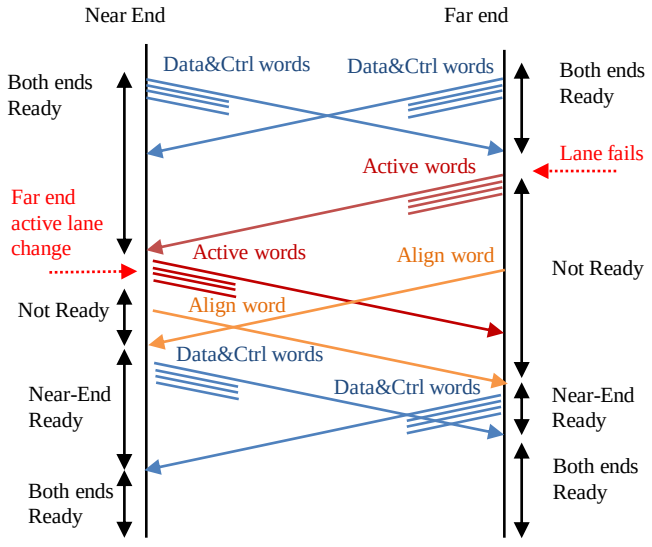


Figure 7. Link ready protocol sequence diagram

	A:TX0	A:TX1	B:RX0	B:RX1	B:Align	B:FIFO0	B:FIFO1	B:RCV0	B:RCV1
269	Active(11)	Active(11)	INIT3	Active(01)		IDLE	Active(01)		RxERROR
270	Active(11)	Active(11)	INIT3	Active(11)		IDLE	Active(11)	RxERROR	
271	Active(11)	Active(11)	IDLE	Align(2,1)		IDLE	Align(2,1)	RxERROR	
272	Active(11)	Active(11)	Active(11)	Active(11)		Active(11)	Align(2,1)	RxERROR	
273	Active(11)	Active(11)	Align(2,0)	Active(11)		Align(2,0)	Align(2,1)	RxERROR	
274	Active(11)	Active(11)	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
275	Align(2,0)	Align(2,1)	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
276	Active(11)	Active(11)	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
277	Active(11)	Active(11)	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
278	Active(11)	Active(11)	Active(11)	Align(2,1)		Active(11)	Active(11)	RxERROR	
279	Active(11)	Active(11)	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
280	Active(11)	Active(11)	Align(2,0)	Active(11)		Align(2,0)	Align(2,1)	RxERROR	
281	Active(11)	Active(11)	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
282	Align(2,0)	Align(2,1)	Active(11)	Active(11)		Active(11)	Active(11)		
283	SDF	SDF	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
284	Data 0	Data 1	Active(11)	Active(11)		Active(11)	Active(11)	RxERROR	
285	EDF 0	EDF 0	Active(11)	Align(2,1)		Active(11)	Active(11)	RxERROR	
286	SDF	SDF	Active(11)	SDF		Active(11)	Active(11)	RxERROR	
287	Data 2	Data 3	Align(2,0)	Data 1		Align(2,0)	Align(2,1)	RxERROR	
288	Data 4	Data 5	SDF	EDF 0		SDF	SDF	RxERROR	
289	Data 6	Data 7	Data 0	SDF		Data 0	Data 1		
290	Data 8	Data 9	EDF 0	Data 3		EDF 0	EDF 0	SDF	
291	Data 10	Data 11	SDF	Data 5		SDF	SDF	Data 0	Data 1
292	Data 12	Data 13	Data 2	Data 7		Data 2	Data 3	EDF 0	
293	Data 14	Data 15	Data 4	Data 9		Data 4	Data 5	SDF	
294	Data 16	Data 17	Data 6	Data 11		Data 6	Data 7	Data 2	Data 3
295	Data 18	Data 19	Data 8	Data 13		Data 8	Data 9	Data 4	Data 5
296	EDF 1	EDF 1	Data 10	Data 15		Data 10	Data 11	Data 6	Data 7
297	SDF	SDF	Data 12	Data 17		Data 12	Data 13	Data 8	Data 9

Figure 8. Simulator tool screenshot of the alignment process

The time it takes for a multi-lane link to resume sending data after a lane has failed or a new lane has been added is less than a few microseconds. More precisely, it is the round trip delay of the ACTIVE words plus the delay between the sending of ALIGN words, which is the time needed to send 8 words.

Fig 8 is a screenshot of the simulator tool used for the validation of the alignment process and the associated state machine. The first set of two columns show the words sent by the sender side using two lanes. The ALIGN word indicates that two lanes are used for sending and the lane numbers are 0 and 1. The ACTIVE word indicates that lane zero and one are active (bits zero and one are set). The second set of two columns show how the row sent is received disaligned at the receiver side. The middle Align column shows in yellow when the state machine is in Not Ready state. The third set of two columns shows how row alignment is achieved using a FIFO and the ALIGN word. Finally the last set of two columns show the words received by the Data Link layer.

D. Unidirectional lanes

Single-lane SpaceFibre implementations must be bidirectional even if the end-user data flow is unidirectional, because feedback from the receiver side is required for the protocol to operate. However in a multi-lane implementation, one lane is enough for the protocol related information and the other lanes can be unidirectional, saving power and mass.

The lane layer initialisation state machine was designed for a bidirectional lane, however some additional rules can be defined to allow a unidirectional lane to reach active state without affecting the operation for bidirectional lanes. The state machine just needs to know if the lane is receive or transmit only and if the far end has the lane in active state.

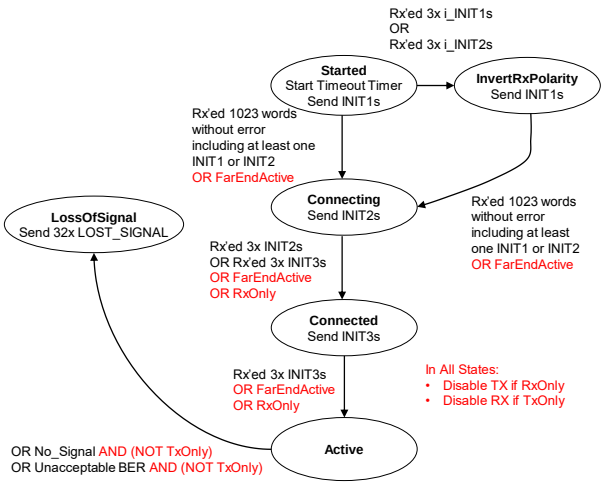


Figure 9. Changes to the lane initialisation state machine

Fig 9 shows in red the required modifications to the lane initialisation state machine. First, if the lane is receive only, the RxOnly condition is set and the state machine immediately moves to Active after reaching the Connecting state. Second, if an ACTIVE word is received after the lane is started indicating that the lane is active at the far end, the FarEndActive condition is set, and the state machine moves to Active. Finally, the LossOfSignal state is not reachable if the lane is TxOnly.

Fig 10 shows how a unidirectional lane is initialised. The side configured as RxOnly, receives INIT1 words until it reaches Connecting state. It then immediately moves to Active. Then, the bidirectional lane(s) send ACTIVE words, which when received, sets the FarEndActive flag. This moves the TxOnly initialisation state machine to ACTIVE.

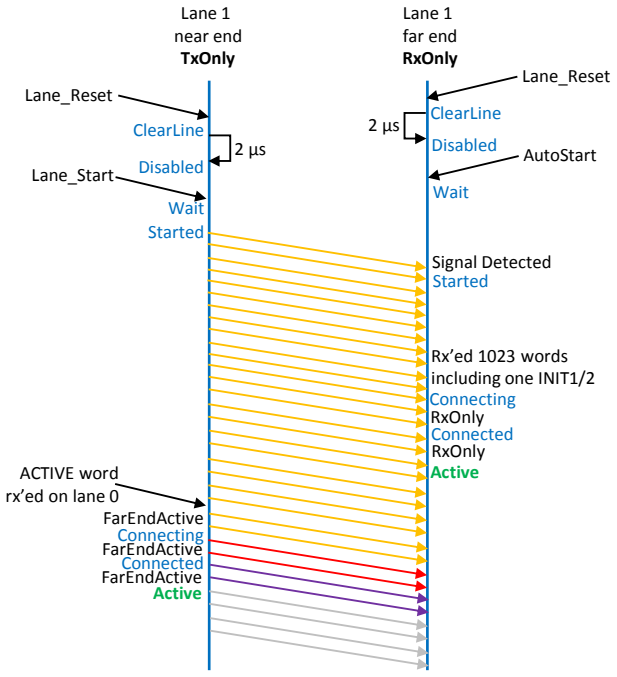


Figure 10. Initialisation of a unidirectional lane

The TxOnly side can not detect loss of signal or receive control words. A mechanism has to be defined in TxOnly lanes to exit the Active state when the RxOnly far end is not anymore in Active state. The solution is to reset the TxOnly lane when ACTIVE words are received in bidirectional lanes indicating that the far end is not anymore active, i.e. the FarEndActive flag is deasserted.

One requirement for unidirectional lanes states that it must be possible for a bidirectional lane to become unidirectional in order to save power. More precisely, a bidirectional lane can be set as RxOnly lane by the user when the data rate sent is reduced. A mechanism is required for the far end to detect this event and change from a bidirectional lane to a TxOnly lane, so it matches the RxOnly setting at the near end.

The solution is for the TxOnly flag to be set when the lane initialisation state machine is in Started state and the FarEndActive flag is set. The FarEndActive flag will move the state machine to Active and the lack of signal at the receiver will be ignored as the TxOnly flag will be set.

E. Hot redundant lanes

An important requirement is the decoupling between the link bandwidth provided by the number of active lanes and the maximum data rate of the end user interface. There are two possibilities:

a) The available link bandwidth is lower than the user interface. This can occur if one or more lanes fails or are disabled. The user interface flow control will limit the data rate of the user.

b) The available link bandwidth is higher than the user interface. This can be useful to provide hot redundancy.

In order to simplify the implementation of the second scenario, the concept of hot redundant lanes is introduced. Hot redundant lanes are lanes that are initialized in the same way than a normal lane, but only send Lane Layer and Multi-Lane layer control words and do not send any Data Link layer word. When no control words must be sent, they send a PRBS sequence that is generated in the same way than the PRBS data words of Idle frames. This mechanism ensures that the word transfer rate between the Multi-Lane layer and the Data Link layer does not exceed the maximum user interface data rate.

Hot redundant lanes must have lane numbers higher than the other lanes. The receiver can identify a hot redundant lane by the content of the ALIGN word received. An ALIGN word sent by a hot redundant lane has the LANES and the iLANES fields both set to zero, which can not occur for non redundant lanes. Hot redundant lanes identified by the receiver are not considered for the reception of Data Link layer words.

F. New control word fields

The addition of multi-lane capabilities requires two new fields in existing control words that do not break compatibility with single-lane implementations:

a) FCT multiplier: Allows to reduce the number of FCTs sent when the data frame size is increased.

b) Multi-Lane capable flag: provided in the INIT3 control word to indicate that the lane is part of a multi-lane link.

IV. Hardware Implementation

After the new specifications that enable multi-lane capabilities to SpaceFibre were successfully simulated in software, a hardware prototype was built using commercial off-the-shelf (COTS) and space-qualified FPGAs. The new multi-lane capable STAR-Dundee SpaceFibre IP Core is an optimised and improved version.

A. Hardware Prototypes

The multi-lane specifications were first evaluated using the STAR-Dundee SpaceFibre PXI board, which has a set of flexible interface connectors that can be used to customise the board, such as SpFi, SpW and external triggers, etc [7].

Fig 11 shows a multi-lane link using two PXI boards with one bidirectional lane and two unidirectional lanes. Each connector has two activity LEDs. If the upper LED is red it indicates that the receiver is disabled. If the lower LED is red it indicates that the transmitter is disabled. Blue colour indicates data transfer. In addition to the SATA laboratory cables used for SpaceFibre, there are two SpaceWire blue cables used for device configuration.

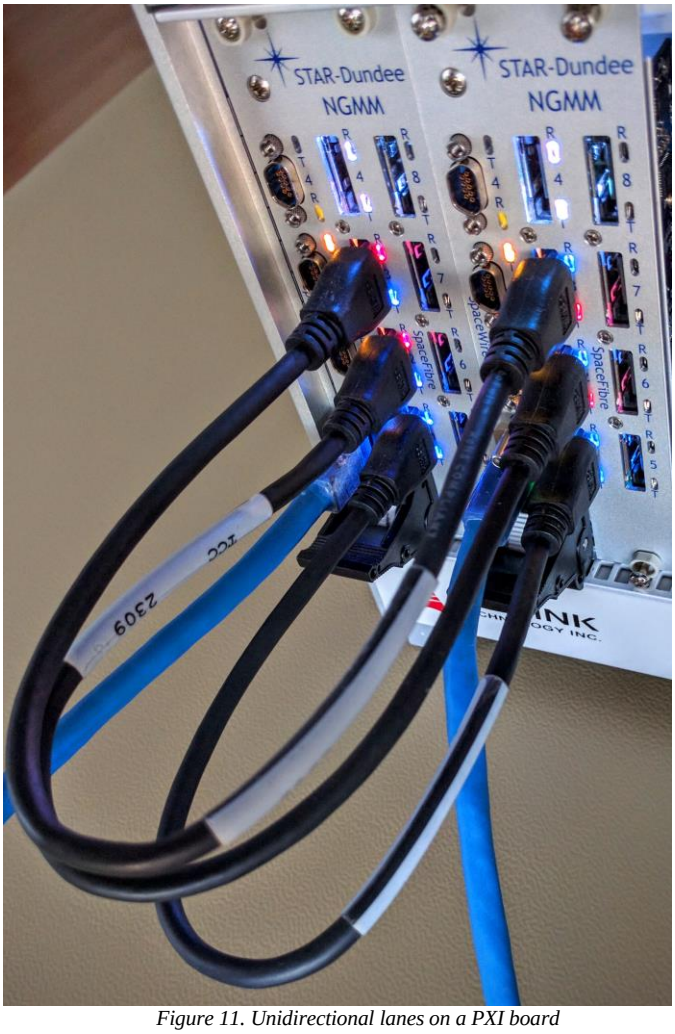


Figure 11. Unidirectional lanes on a PXI board

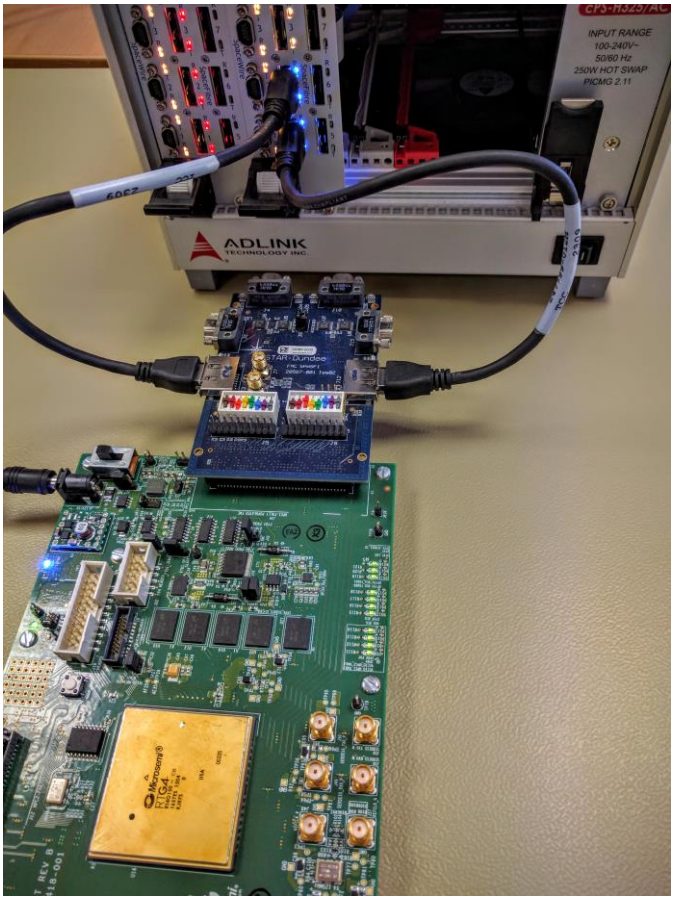


Figure 12. RTG4 development board with a multi-lane SpaceFibre link

The design was then optimised and ported to the radiation hardened RTG4 FPGA. Fig 12 shows the RTG4 development board with a multi-lane link connected to a PXI board using two bidirectional lanes.

For the validation of the new protocol capabilities, the STAR-Fire software was updated to support the new multi-lane capable features. Fig 13 is a screenshot of the the STAR Fire Analyser view. The two middle columns shows the words being sent by lane numbers 1 (left) and 0 (right). At each side the word is decode in its symbol components. The analyser was triggered on the event of the first ACTIVE word sent after the link was started with three lanes. As stated, ACTIVE words are sent when a lane becomes active until alignment is achieved.

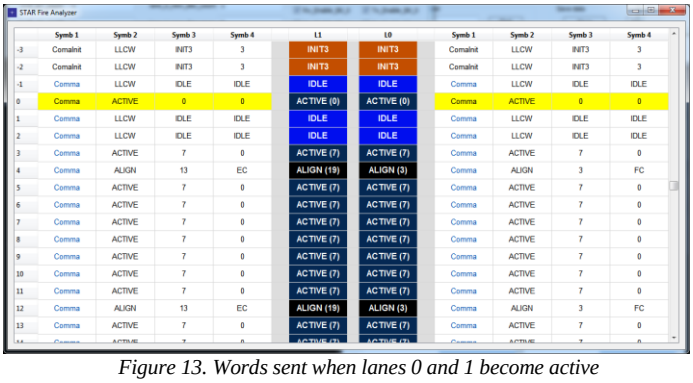


Figure 13. Words sent when lanes 0 and 1 become active

The left side of Fig 14 shows what happens later when alignment is achieved and the alignment state machine moves from Not Ready to Near-End Ready. It is allowed then to send Data Link layer words such as the FCT control words. The ACTIVE words indicate that the first three lanes are active (7_{16}, 111_2). The ALIGN word indicates that three lanes are used for sending and the lane number of each lane (19_{10}, 13_{16}). The right side shows sometime later when data frames are sent with user data from virtual channel 1. At this time, the alignment state machine is in Both Ends Ready as no ALIGN words are being sent.

Figure 14. Words sent for alignment (left) and sending data (right).

Fig 15 shows a link with two lanes in which lane 1 fails and starts sending LOS control words before disabling the SerDes. The alignment state machine moves to Not Ready and ACTIVE words sent indicate lane 1 is not anymore active. After re-alignment, a NACK control word followed by a RETRY control word are sent, so both ends can resume sending data.

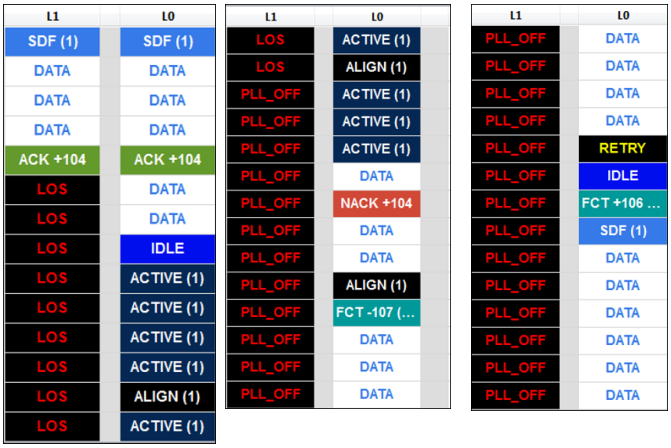


Figure 15. Words sent when a lane fails (from left to right)

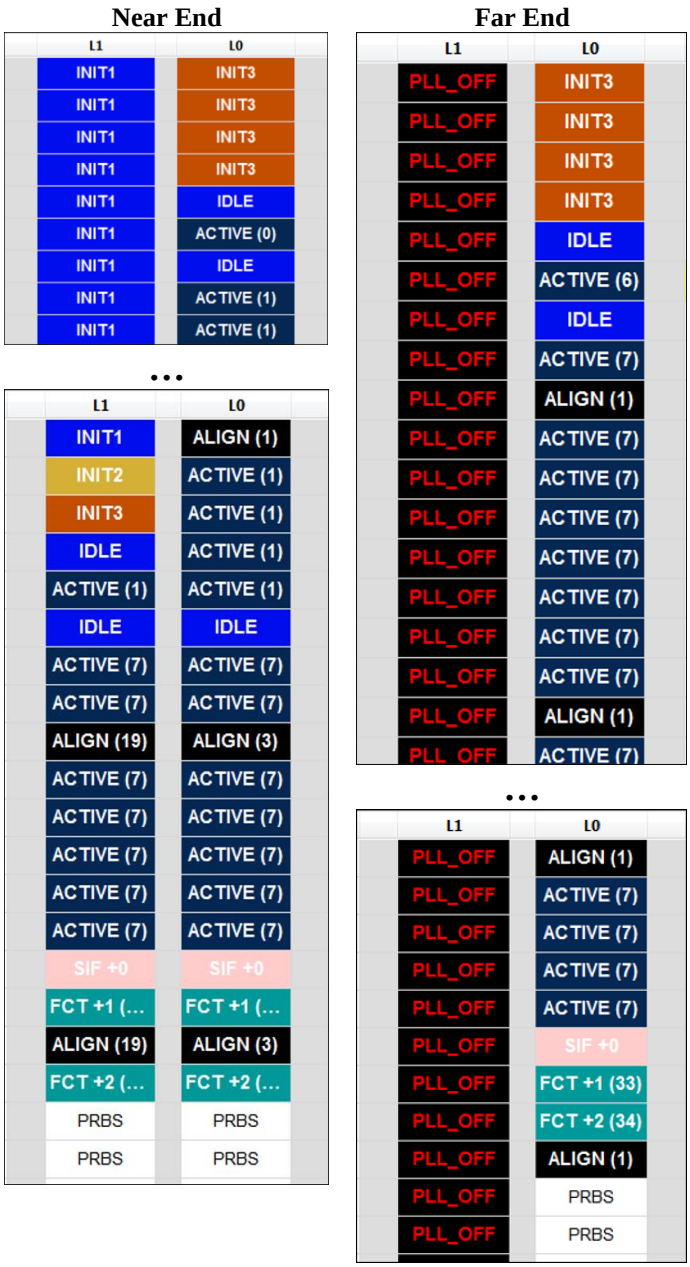


Figure 16. Link starts with lane 1 unidirectional with TxOnly set

Fig 16 shows a multi-lane link with three lanes in which lane 1 is a unidirectional transmit-only lane. That is why the far end has the SerDes transmitter of lane 1 disabled (PLL_OFF). In the near end, this lane starts sending INIT1s until the far end achieves Active state and sends ACTIVE words indicating this lane is active. The near end sends INIT2 and INIT3 and reaches active state on the reception of these ACTIVE words.

B. STAR-Dundee SpaceFibre IP Core

The STAR-Dundee SpaceFibre IP Core was updated to support multi-lane capabilities after the hardware implementations were successfully validated and optimised for low resource usage and easy of use.

Table I provides the resource usage for two radiation hardened FPGAs, Microsemi RTG4 and Xilinx Virtex-5QV,

for different number of lanes and virtual channels. Lanes can operate up to 3.125 Gbps.

TABLE I. RESOURCE USAGE

	RTG4			Virtex-5QV		
	LUT	DFF	RAM Block	LUT	DFF	RAM Block
2 Lanes 1 VC	6494 4.3%	5351 3.5%	8 3.8%	3858 4.7%	3938 4.8%	8 2.7%
2 Lanes 2 VC	7314 4.8%	6088 4.0%	12 5.7%	4503 5.5%	4382 5.3%	12 4.0%
3 Lanes 2 VC	8997 5.9%	7413 4.8%	12 5.7%	5416 6.6%	5226 6.4%	12 4.0%

The IP Core has been designed to fully support the redundancy capabilities of multi-lane. When using hot redundancy, the data flow of the user is not affected when a lane fails, as the data is internally buffered during the time it takes to resume sending data, which is less than 2 µs. When not using a hot redundant lane, there is a graceful degradation of link bandwidth and the QoS mechanism ensures that most important data is sent first. If a redundant lane is available it will be activated in less than 20 µs, providing warm redundancy.

Fig 17 shows the floorplan of a Virtex-5QV with the IP core constrained to be placed in one of the tiles. Using the Xilinx transceiver capabilities, the IP Core can work with a single clock input signal. The user can write and read data to/from the IP Core with the AXI4-Stream interface, using any other clock frequency as the IP includes synchronisation buffers.

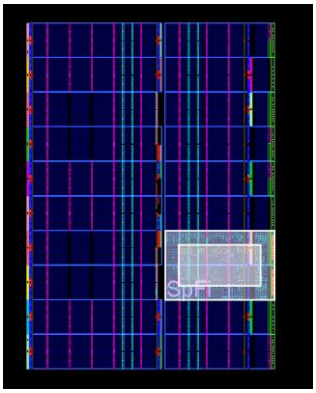


Figure 17. STAR-Dundee multi-lane IP Core in Virtex-5QV

V. CONCLUSION

The new SpaceFibre multi-lane capabilities increase dramatically the data throughput of SpaceFibre links to meet the requirements of next generation of spacecraft payloads.

With the designed multi-lane layer, the additional lanes can also provide hot or warm redundancy, and graceful degradation of the link bandwidth when no redundant lanes are available. In the event of a lane failure, the link is again operative in just a few microseconds, which is close to the round trip delay of the lane, without user intervention and without any data loss.

Furthermore, the flexibility in the number of lanes of a multi-lane link and the support of unidirectional lanes, allows for significant savings in mass and power, which are critical in space applications.

The multi-lane specifications have been validated in simulation and hardware prototypes. These specifications have been designed to be easy to implement in slower radiation hardened FPGAs. The STAR-Dundee SpaceFibre IP Core has been updated to provide all these new multi-lane capabilities in RTG4 and Virtex-5QV FPGAs with low resource usage and high performance.

ACKNOWLEDGMENT

The research leading to these results has received funding the European Space Agency under ESA contract numbers 4000102641 and from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 263148 and 284389.

REFERENCES

[1] Next Generation Processor for On-board Payload Data Processing Application ESA Round Table Synthesis, ESA, TEC-EDP/2007.35/RT, October 2007.

[2] S. Parkes, A. Ferrer, A. Gonzalez and C. McClements, "SpaceFibre Standard", Draft H4, April 2016, available from https://indico.esa.int/indico/event/126/session/0/contribution/1 (last accessed 29th August 2016).

[3] T. Scheckel, "Serial RapidIO: Benefiting system interconnects", Proceedings - IEEE International SOC Conference, pp. 317-318, 2005.

[4] J. Ajanovic, "PCI express 3.0 overview", 2009 IEEE Hot Chips 21 Symposium (HCS), Stanford, CA, 2009, pp. 1-61.

[5] "Interlaken Protocol Definition", A joint specification of Cortina Systems and Cisco System, Revision 1.2, Cortina Systems Inc. and Cisco Systems Inc., 7 October 2008.

[6] Y. Otake, "The study and proposal for improvement the multi-lane operation of SpaceFibre protocol", SpaceWire Conference 2014, Athens.

[7] A. Gonzalez, "A new Generation of SpaceFibre Test and Development", SpaceWire Conference 2016, Yokohama.

SpaceFibre Flight Equipment

SpaceFibre, Long Paper

Steve Parkes, Albert Ferrer Florit,
Alberto Gonzalez Villafranca, Chris McClements,
Bruce Yu, Pete Scott, Julie Logan,
STAR-Dundee Ltd.,
STAR House, 166 Nethergate, Dundee, DD1 4EE, UK
steve.parkes@star-dundee.com

David McLaren,
Space Technology Centre, University of Dundee,
166 Nethergate, Dundee, DD1 4EE, UK

***Abstract*— SpaceFibre is a new standard for spacecraft on-board data-handling networks, which runs over both electrical and fibre optic media. It provides high bandwidth, low latency, fault recovery and novel QoS that combines priority, bandwidth reservation and scheduling. SpaceFibre is backwards compatible with SpaceWire at the network level, allowing existing SpaceWire equipment to be incorporated into a SpaceFibre network without modification. SpaceFibre is now being designed into its first spaceflight missions. This paper describes SpaceFibre flight equipment being designed by STAR-Dundee for space flight applications. This includes a range of SpaceFibre IP cores targeted at radiation tolerant FPGAs and the SpaceFibre interfaces in a radiation tolerant many core DSP processor.**

***Index Terms* — SpaceFibre, SpaceWire, Flight Equipment, Networking, Spacecraft Electronics.**

I. Introduction

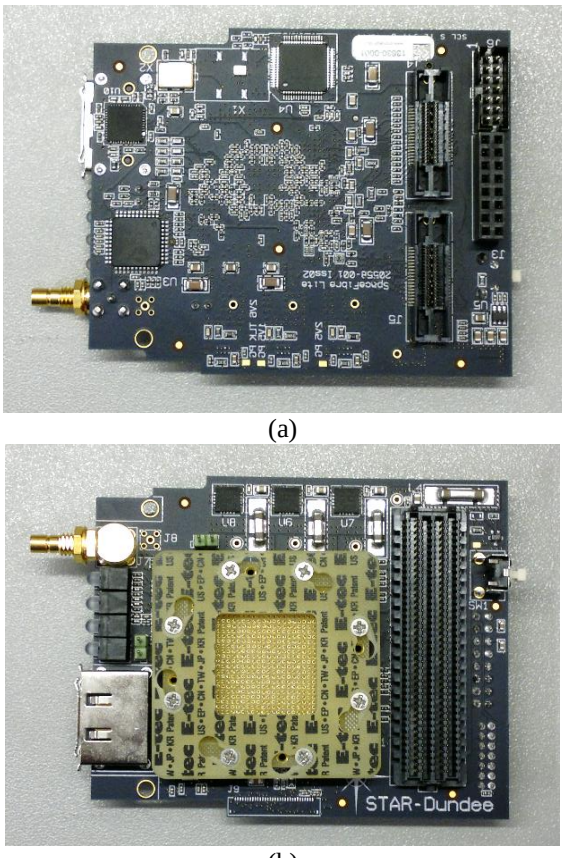
SpaceFibre [1][2][3] is a new standard for spacecraft on-board data-handling networks, initially designed to deliver multi-Gbit/s data rates for synthetic aperture radar and high-resolution, multi-spectral imaging instruments. The addition of quality of service (QoS) and fault detection, isolation and recovery (FDIR) capabilities to SpaceFibre has resulted in a unified network technology. SpaceFibre provides high bandwidth, low latency, fault isolation and recovery suitable for space applications, and novel QoS that combines priority, bandwidth reservation and scheduling and which provides babbling node protection [4]. SpaceFibre is backwards compatible with the widely used SpaceWire standard [5] at the network level allowing simple interconnection of existing SpaceWire equipment to a SpaceFibre link or network.

This paper describes SpaceFibre equipment being designed by STAR-Dundee for space flight applications. This includes a range of SpaceFibre IP cores targeted at radiation tolerant FPGAs, the SpaceFibre interfaces in a radiation tolerant many core DSP processor and boards, subsystems and instrument processing units, containing these devices.

II. SpaceFibre Interface in the RTAX FPGA

A version of the SpaceFibre IP core targeted for high performance and small size in flight qualified FPGAs is currently being developed by STAR-Dundee Ltd. This IP core is designed to support instrument interfacing with SpaceFibre using existing flight proven FPGAs and SerDes devices. It is expected that this design will reduce the size of the SpaceFibre IP core for instrument interfaces significantly.

A board that implements this "SpaceFibre-Lite" IP core in a Microsemi AX1000 FPGA is illustrated in Figure 1.



(a)

(b)

Figure 1. SpaceFibre-Lite board for Microsemi AX1000 FPGA; (a) top-side and (b) bottom-side

On the bottom side of the SpaceFibre-Lite board is a socket for an AX1000 FPGA, which is the commercial equivalent of the radiation tolerant RTAX1000 FPGA [6]. This FPGA does

not include a SerDes so an external SerDes device is required. Texas Instruments have a suitable radiation tolerant SerDes device: the TLK2711-SP Wizard Link device [7]. This device contains both a transmitter and receiver and offers data rates from 1.28 to 2.0 Gbits/s (1.6 to 2.5 Gbits/s data signalling rates). The transmitter takes in 16-bit wide serial data, encodes it using 8B/10B encoding and serialises it for transmission over a differential signal pair. The receiver takes the serial data, de-serialises it, and performs 8B/10B decoding to provide the 16-bit parallel data. The TLK2711A (commercial version) can be seen on the top-side of the board, at the top of Figure 1.

The SpaceFibre-Lite interface has two virtual channels and a broadcast message interface. One virtual channel is used for sending or receiving high data-rate application data, which requires substantial link bandwidth. The other virtual channel is used for receiving configuration, control and housekeeping requests from a remote computer and for returning status and housekeeping information. This latter virtual channel is typically set to high priority, but uses little bandwidth.

The SpaceFibre-Lite board has an FMC connector for connecting to a host system, e.g. another FPGA development board. A 32-bit interface is provided on this FMC connector for sending and receiving data between the host system and the SpaceFibre virtual channels in the AX1000 FPGA. This interface can also be used for configuring the SpaceFibre interface and for accessing the broadcast message interface. For test purposes, a pair of Mictor connectors are provided on the parallel interface to the FMC connector for connection to a logic analyser.

The SpaceFibre serial interface is connected to an eSATA connector which is used in SpaceFibre electrical ground support equipment. This connector can be seen on the bottom left of Figure 1 (b).

All of the major components on the SpaceFibre-Lite board are commercial equivalents of radiation tolerant, spaceflight grade components. It operates at a data signalling rate of 2.5 Gbits/s and demonstrates that SpaceFibre is at TRL5, ready to fly.

III. SpaceFibre in the RTG4 FPGA

The Microsemi RTG4 is a new generation radiation tolerant FPGA [8]. It has extensive logic, memory, DSP blocks, and IO capabilities and is inherently radiation tolerant, having triple mode redundancy built in. The RTG4 has a flash configuration memory built into the device. In addition the FPGA incorporates 16 SpaceWire clock-data recovery circuits and 24 multi-Gbits/s SerDes lanes to support high-speed serial protocols like SpaceFibre. The integrated radiation tolerant SerDes make the RTG4 ideal for the implementation of SpaceFibre.

A SpaceFibre interface has been implemented in the RTG4 FPGA and tested extensively [9]. The test design incorporates two SpaceFibre interfaces and four SpaceWire interfaces. One SpaceFibre interface has eight virtual channels and the other has four. These two SpaceFibre interfaces are connected back to back with VC4-7 on one interface connected to VC4-7 on the other interface. The four SpaceWire interfaces are connected to VC0-3 on the SpaceFibre interface with eight virtual channels. This is illustrated in Figure 2.

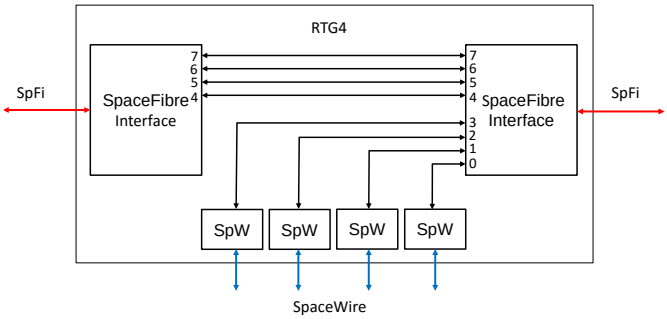


Figure 2. Functional block diagram showing interconnection between SpaceFibre and SpaceWire interfaces in an RTG4 FPGA

The design is implemented on the Microsemi RTG4 development board with SpaceWire and SpaceFibre connectors provided via an FMC board, as shown in Figure 3.

Figure 3. Microsemi RTG4 development board used to test the SpaceFibre interface

The SpaceFibre interfaces operate at up to 3.125 Gbits/s. A multi-lane interface has also been implemented in the RTG4 and validated [10]. The multi-lane IP core has two-lanes with a third lane available in hot or cold standby.

The SUNRISE SpaceFibre routing switch [11] is currently being transferred to the RTG4 FPGA, using a specially designed board [12], which is shown in Figure 4.

Figure 4. Prototype board for SUNRISE SpaceFibre Routers

IV. SpaceFibre Multi-Lane Interface in the RTG4 FPGA

The multi-laning capabilities of the SpaceFibre protocol allow several lanes to operate in parallel to provide enhanced throughput [10]. For example, with four lanes running at 2.5 Gbits/s each an aggregate throughput of 10 Gbits/s is achieved. SpaceFibre multi-laning can operate with any number of lanes, from 1 to 16. Each lane is normally bi-directional, but to support spaceflight instruments with very high-data rate in one direction and to save mass and power, it is possible to have some uni-directional lanes in a multi-lane link, provided that at least one lane is bi-directional. SpaceFibre multi-laning also supports graceful degradation in the event of a lane failure. If a lane fails, the multi-lane link will rapidly reconfigure to use the remaining lanes so that important (high priority) information can still get through. It takes a couple of microseconds for this reconfiguration to occur, which happens without loss of information. Clearly, with reduced bandwidth some information will not be sent over the link, but this will be less important, low priority, information. If a redundant lane is available in the link, it can be enabled and full capacity operation will resume.

Figure 5. Demonstration of SpaceFibre Multi-Laning

The photograph in Figure 5 shows a demonstration of the multi-laning capability of SpaceFibre. A four lane link was demonstrated with low-priority, high-bandwidth traffic flowing over some virtual channels and high-priority video data over another virtual channel. Lanes were unplugged with corresponding loss in bandwidth, but the link continued to operate sending the "critical" video data without interruption. Only when all four lanes were unplugged, did the video data stream cease. As soon as any of the four lanes were plugged back in, the video stream continued once more.

V. SpaceFibre Engineering Model

STAR-Dundee is currently designing a flight Engineering Model level board for the RTG4 which will support SpaceWire and SpaceFibre applications. The architecture of this board is illustrated in Figure 6.

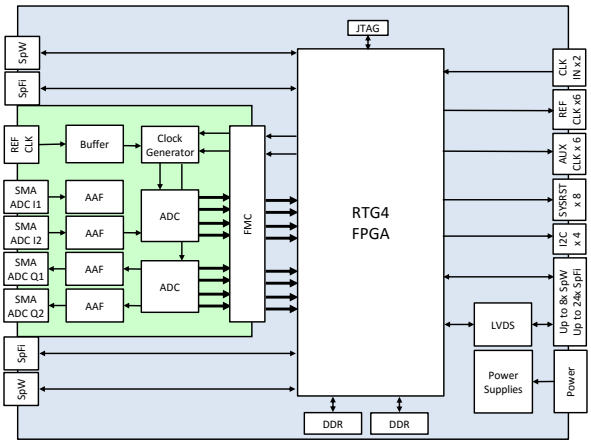


Figure 6. SpaceVPX-RTG4 Board Block Diagram

The SpaceVPX-RTG4 board is a 3U board designed to conform to the emerging VITA78.1 SpaceVPX-Lite standard [13]. The main component on the board is the RTG4 FPGA (PROTO Silicon). It is connected to two independent banks of DDR memory, each supporting Error Detection and Correction (EDAC). Two SpaceWire and two SpaceFibre interfaces are provided on the front panel. The SpaceVPX-Lite backplane supports a SpaceWire control plane and a SpaceFibre data plane along with standard utility plane functions. An FMC type daughterboard connector allows connection to various daughterboards. A dual, 3 Gsamples/s ADC FMC board is available supporting demanding DSP applications. Other daughter boards are planned. The board is conduction cooled.

The board can be configured to operate as a SpaceVPX-Lite System Controller or as a versatile SpaceVPX-Lite Payload Processing board. The System Controller incorporates an ARM Cortex M1 processor running in the FPGA, and has two SpaceWire and two SpaceFibre interfaces on the front panel. It provides the VITA78.1 radial REF_CLK and AUX_CLK signals to each of up to six Payload boards. It can provide either SpaceWire or SpaceFibre radial control plane connections to each Payload board. These control plane interfaces also provide the Payload management function using RMAP [14]. The System Controller is designed to operate in a dual redundant configuration with control plane cross strapping to each Payload board. Cold sparing of the RTG4 is addressed in the board design.

The SpaceVPX-RTG4 board can also act as a Payload board, with control plane connections to each of the two system controller boards. Data plane connections are provided on the board to support full mesh interconnection between the six payload boards.

The components on the board are commercial equivalents of flight grade components.

This board is currently being used to implement the engineering model of a wideband spectrometer for a THz radiometer instrument, being developed in the UK [15]. When fitted with the ADC FMC board each SpaceVPX-RTG4 board will be able to process 1-2 GHz bandwidth signals into 1-5MHz spectral components. The design of the FFT processor is

currently underway based on previous designs implemented and tested in Xilinx Virtex 5 FPGAs.

At present the board is in the PCB layout stage. A physical model of the SpaceVPX-RTG4 board is illustrated in Figure 7.

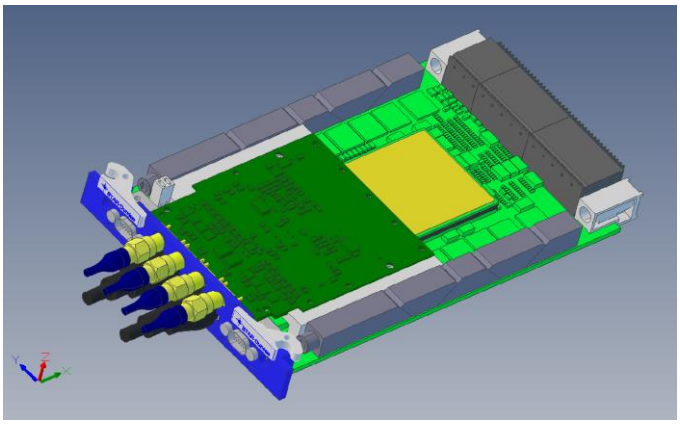


Figure 7. SpaceVPX-RTG4 Physical Model

SpaceVPX-Lite Power Switches and Power Supply modules are also under development along with a backplane and conduction cooled rack.

VI. Ramon Chips RC64

Ramon Chips are developing a many core DSP processing chip in radiation tolerant technology. The RC64 [16], is a novel rad-hard 64-core digital signal processing chip, with a performance of 75 MACS, 150 GOPS and 38 GFLOPS (single precision) and low power consumption, dissipating less than 10 Watts. The RC64 integrates sixty-four advanced DSP cores, a hardware scheduler, 4 MBytes of multi-port shared memory, a DDR2/DDR3 memory interface, and twelve 3.125 Gbps full-duplex, high-speed SpaceFibre serial links, four of which can also support serial Rapid IO.

The RC64 architecture is illustrated in Figure 8. A central scheduler assigns tasks to processors. Each processor executes its task from its cache storage, accessing the on-chip 4MByte shared memory only when needed. When task execution is done, the processor notifies the scheduler, which subsequently assigns a new task to that processor. Access to off-chip streaming channels, DDR2/DDR3 memory, and other interfaces happens only via programmable DMA channels. This approach simplifies software development and it is found to be very useful for DSP applications, which favour streaming over cache-based access to memory. Hardware events, asserted by communication interfaces, initiate software tasks through the scheduler. This enables high event rates to be handled by the many cores efficiently.

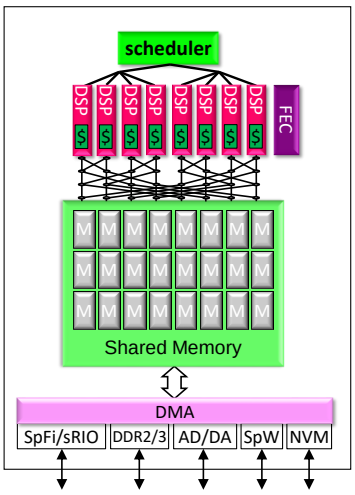


Figure 8. RC64 Many Core DSP Processor Block Diagram (only 8 DSP processors are shown)

The RC64 is implemented as a 300 MHz integrated circuit on a 65nm CMOS technology, assembled in a hermetically sealed ceramic CCGA624 package and qualified to the highest space standards. Supported communication applications include frequency multiplexing, digital beam forming, transparent switching, modems, packet routing and higher-level processing. The 12 SpaceFibre interfaces on the RC64 were designed by STAR-Dundee.

STAR-Dundee is currently designing a SpaceVPX-Lite board containing an RC64 many core DSP processor. A block diagram of this board is shown in Figure 9.

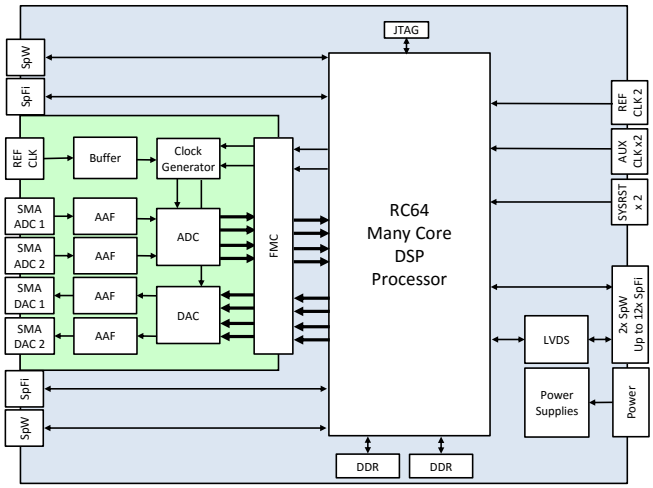


Figure 9. SpaceVPX-RC64 Board Block Diagram

The SpaceVPX-RC64 board contains an RC64 DSP processor attached to DDR memory. The board is designed as a SpaceVPX-Lite Payload board. It receives nominal and redundant REFLCK, AUXCLK and SYSRST signals from the backplane. These signals originate from the nominal and redundant System Controller boards. Nominal and redundant control plane interfaces are also provided from the System

Controller boards via the backplane connectors. The control plane can be either SpaceWire or two-lane SpaceFibre.

There are a pair of SpaceWire interfaces and a pair of SpaceFibre interfaces on the front panel. An FMC connector on the board allows for the connection of a range of FMC type boards to be added. All components on the board are commercial equivalents of radiation tolerant parts. Like the SpaceVPX-RTG4 board the SpaceVPX-RC64 board is conduction cooled.

VII. SpaceFibre Interface Chip

The SpaceFibre ECSS standard is close to being published and SpaceFibre is already being considered for several space missions. There is a need for a range of radiation tolerant SpaceFibre chips to support the missions that plan to use this technology. STAR-Dundee has won a contract from ESA to develop such a device, which is able to meet the instrument interface and avionics equipment requirements for high-speed serial links. This design will build on the extensive experience that STAR-Dundee has with SpaceFibre and in particular on the experimental SpaceFibre interface device designed by STAR-Dundee with European Commission Framework 7 research funding [17].

Conclusions

SpaceFibre is a new generation of the widely used SpaceWire spacecraft on-board data-handling network technology, which has over ten times the performance (per lane) and operates over electrical or fibre optic media. Integrated quality of service and fault detection, isolation and recovery mechanisms enable SpaceFibre to be used for guidance and navigation control, time-distribution, event signalling, command and control, as well as very high data-rate payload data-handling, all with a single, unified network. This reduces cost, mass and risk, improves reliability and simplifies redundancy.

STAR-Dundee has developed a range of SpaceFibre IP cores for spaceflight applications including a single-lane and multi-lane interface targeted for the Microsemi RTG4 and Xilinx Virtex-5QV FPGAs. A SpaceFibre routing switch IP core for the RTG4 is currently under development. STAR-Dundee's IP cores are also being used in a range of radiation tolerant ASIC devices including the Ramon Chips RC64 many core DSP processor and the ESA SpaceFibre Interface Chip. A range of engineering model level boards is being designed by STAR-Dundee based on the emerging VITA 78.1 SpaceVPX-Lite standard. This equipment is targeted at a range of spaceflight signal and image processing applications and is already being designed into the UK LOCUS TeraHertz sounder instrument.

Acknowledgments

The research leading to these results has received funding from the European Space Agency under ESA contract numbers 4000102641 and 17938/03/NL/LvH, from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement numbers 263148 and 284389, and from the UK Space Agency and CEOI-ST under University of Leicester contract numbers: RP10G0348A02, RP10G0348B206 and RP10G0348A207.

References

[1] S. Parkes, A. Ferrer Florit and A. Gonzalez Villafranca, "SpaceFibre Standard", Draft H5, University of Dundee, July 2016.

[2] S. Parkes, C. McClements and M. Suess, "SpaceFibre", International SpaceWire Conference, St Petersburg, Russia, 2010, ISBN 978-0-9557196-2-2, pp 41-45.

[3] S. Parkes, A. Ferrer, A. Gonzalez, & C. McClements, "SpaceFibre: Multiple Gbits/s Network Technology with QoS, FDIR and SpaceWire Packet Transfer Capabilities", International SpaceWire Conference, Gothenburg, June 2013.

[4] S. Parkes et al, "SpaceFibre: Multi-Gigabit/s Interconnect for Spacecraft On-board Data Handling", IEEE Aerospace Conference, Big Sky, Montana, 2015.

[5] ECSS Standard ECSS-E-ST-50-12C, "SpaceWire, Links, Nodes, Routers and Networks", Issue 1, European Cooperation for Space Data Standardization, July 2008, available from http://www.ecss.nl.

[6] http://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/rtax-s-sl

[7] Texas Instruments, "TLK2711A 1.6 TO 2.7 GBPS TRANSCEIVER", SLLS908A, September 2009.

[8] http://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/rtg4

[9] S. Parkes et al, "SpaceWire and SpaceFibre on the Microsemi RTG4 FPGA", IEEE Aerospace Conference, Big Sky, Montana, 2016.

[10] A. Ferrer Florit, A. Gonzalez Villafranca and S. Parkes, "SpaceFibre Multi-Lane", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

[11] S. Parkes, A. Ferrer Florit, A. Gonzalez Villafranca, Chris McClements and David McLaren, "SpaceFibre Networks", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

[12] A. Gonzalez Villafranca, S. Parkes, C. McClements, B. Yu, P. Scott and A. Ferrer Florit, "A New Generation of SpaceFibre Test and Development Equipment", International SpaceWire Conference, Yokohama, Japan, 2016, ISBN 978-0-9954530-0-5.

[13] Scott Goedeke, et al, "SpaceVPXLite, Lightweight SpaceVPX Systems Specification", VITA 78.1, Draft revision 2.3, VITA, 14 July 2016.

[14] ECSS Standard ECSS-E-ST-50-52C, "SpaceWire – Remote memory access protocol", Issue 1, European Cooperation for Space Data Standardization, 5 February 2010, available from http://www.ecss.nl.

[15] S.P. Rea, et al, "The Low-Cost Upper-Atmosphere Sounder (LOCUS)", 26th International Symposium on Space TeraHertz Technology, Cambridge, MA, 16-18 March 2015.

[16] R. Ginosar, P. Aviely, T. Israeli and H. Meirov, "RC64: High Performance Rad-Hard Manycore", IEEE Aerospace Conference, Big Sky, Montana, 2016.

[17] S. Parkes, A. Ferrer-Florit, A. Gonzalez-Villafranca, C. McClements, R. Ginosar, T. Liran, G. Sokolov, G. Burdo, N. Blatt, P. Rastetter, M. Krstic, A. Crescenzio, "A Radiation

SpaceFibre Based On-board Networks for Real-Time Video Data Streams

SpaceFibre, Long Paper

Alexey Khakhulin, Igor Orlovsky

Rocket and Space Corporation Energia after S.P. Korolev
Korolev, Moscow region, Russia
{Alexey.Hahulin, Igor.Orlovsky}@rsce.ru

Yuriy Sheynin, Ilya Korobkov, Valentin Olenev, Elena Suvorova, Irina Lavrovskaya

Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
sheynin@aanet.ru, {ilya.korobkov, valentin.olenev}@guap.ru, suvorova@aanet.ru, irina.lavrovskaya@guap.ru

***Abstract*—High-speed onboard networks for the space industry with a lot of tasks that could be solved only by transmitting large data streams in a short time, with minimum overheads and accepted latencies. Particular tasks for data transmission require various types of traffic and onboard network topologies. Video data in many applications generate high throughput real-time data streams, from most demanding onboard traffic. The SpaceFibre protocol, which gives an ability to transmit data with high speeds and different quality of services (QoS), could be prospective technology for the spacecraft tasks and missions. Implementation of SpaceFibre and considering its application for Russian space missions is going on.**

The paper presents use cases for SpaceFibre based onboard networks for real-time video data streams in prospective missions. We consider features and characteristics of raw, non-compressed video data streams for processing and real-time control (e.g. to support docking), data streams of compressed motion imagery to record video, science experiment high quality video, robotics, high definition television frames to monitors, etc. The paper considers requirements and restrictions for building SpaceFibre onboard networks for real-time video data streams. Streaming Data Transport Protocol is mapped on a SpaceFibre network for transmission of streaming data from onboard cameras (video stream), to onboard monitors and to a high rate downlink.

***Index Terms* — Spacecraft onboard networks, Streaming data, SpaceFibre, Motion imagery.**

I. INTRODUCTION

Modern onboard networks for space industry include a lot of streaming traffic sources. Examples of such sources are video cameras. Its traffic typically has high rate and requires essential part of network resources. In many cases delivery time and jitter of delivery time for video traffic is strongly constrained. Influence of this traffic to other traffic may be dramatical.

Main features of streaming traffic are:

- PDUs have equal size;
- Intervals between sequential PDUs generations are equal;
- PDUs are structurally homogenous
- PDUs arrives continuously
- Loss of an individual PDU is not critical [1].

II. VIDEO DATA STREAMING CHARACTERISTICS

The parameters of video from CCSDS 766.1-B-1 Digital Motion Imagery (hereinafter referred to as CCSDS) standard are presented Table I. It is a standard that identifies which television and video industry standards should be utilized for interoperability in a spacecraft, between spacecrafts and between a spacecraft and Earth. The CCSDS specification describes real-time video data transmission and video streaming (telecasting). Transmitted data can be uncompressed, compressed or encrypted (Secure JPEG2000) [2].

TABLE I. PARAMETERS OF VIDEO TRAFFIC DESCRIBED IN AVIATION STANDARD CCSDS 766.1-B-1

Traffic	Resolution	*Frame size, Kbyte	*Line size, Kbyte	Playback frequency, Hz
Personal video conferencing	320x240..1280 x720	150..1800	0,625..2,5	10 – 60
Medical conferencing	320x240..1280 x720 Standard resolution 640x480	150..1800 600	0,625..2,5 1,25	10 – 60
Situational awareness	640x480..1280 x720	600..1800	1,25..2,5	25 – 60
Public affairs				**24, 25, 60**
High Resolution Digital Imaging	1920x1080..4096x2160	4050..17280	3,75..8	24 – 120

* - all evaluations are done when color depth is 16 bit

III. SHORT INFORMATION ABOUT THE SPACEFIBRE

The layers from the Physical layer until the Network layer are defined in the SpaceFibre protocol.

В текущей версии стандарта SpaceFibre определены уровни протокола до сетевого.

A. Evaluation of data transmission overheads on a SpaceFibre data link

The 8B/10B coding is used at the Lane layer for data transmission via physical link. Correspondingly the useful throughput on this layer is 0,8 (80%) from the physical throughput.

Data are transmitted via data link in frames. The maximal payload size of a frame is 256 bytes of data, FILL or EP symbols. (The SpaceFibre frame can contain one or some SpaceWire packets or parts of consequent packets.)

The size of the frame header and the size of the frame tail is four bytes. The credit mechanism is used for flow control. A receiving side sends credits in accordance with free space in its buffers. One credit (FCT) corresponds to 256 NChars (data bytes, FILLs , EOPs, EEPs). The length of FCT symbol is 4 bytes. The receiving side sends responses for received frames. (ACK response indicates that the frame is received correctly, NACK response indicates any errors.) The length of response symbol is four bytes. To avoid essential overheads of physical channel by responses, one response may be sent for some sequential frames with small length. However, if a frame has the maxima length the response is sent for every frame.

Thus for the useful throughput evaluation we suppose that one ACK and one FCT correspond to every data frame. These ACK and FCT are transmitted in opposite direction than data frame, therefore they influence to useful throughput only when data traffic is transmitted in both directions.

Accordingly to this, if data are transmitted in one direction and frames with maximal size are used for data transmission, the useful throughput of data link will be 97% from throughput of Lane layer, and, correspondingly, 77% from throughput of the physical channel.

If data are transmitted in both directions the useful throughput of data link will be 94% from throughput of Lane layer, and, correspondingly, 75% from throughput of physical channel.

If frames with smaller size are used for data transmission, the useful throughput would be less.

B. QoS at the Data link layer

The QoS mechanisms in SpaceFibre standard are supported at the Data link layer:

- Priorities
- Reserved bandwidth
- Scheduling
- Guaranteed delivery

Virtual channels are used for QoS implementation. The Data link layer may support up to 32 virtual channels. Particular implementations may support less quantity of virtual channels, because hardware cost of this mechanism is essential.

The priority layer, reserved bandwidth and set of timeslots for data transmission should be assigned for every virtual channel.

These parameters and QoS mechanisms determine sequence of data frames transmission to physical channel for different virtual channels.

The priority mechanism. The priority level should be assigned to every virtual channel. Several virtual channels may have same priority level or every virtual channel can have uncial priority. If several virtual channels have data for transmission (and credits) the first will be transferred a frame from the virtual channel with highest priority. When several virtual channels have the same (and highest) priority selection will be made by other QoS parameters.

Priorities of virtual channels are used only at the data link layer. In general case they do not correlate with packets' priorities at the network layer.

Reserved bandwidth. The portion of channel's throughput should be reserved for every virtual channel. This portion should include not only data payload but overheads also (SpaceFibre frame header and tail, ACK|NACK, FCT).

The virtual channel may use more bandwidth than reserved when other channels with highest and same priority do not have any data for transmission. It can use all channel bandwidth during long time (implementation dependent) to the prejudice of the channels with lower priorities.

Selection of next virtual channel for data transmission is made after transmission of every frame. Frame borders may be not match with the packet's borders. Thus after transmission via physical channel of a part (in a frame) belongs to one packet, the a part of another packet (transmitted via other virtual channel) will could be transmitted. Thus transmission of rare part that belongs to the first packet may be essentially delayed.

Selection of next virtual channel for data transmission is made after transmission of every frame. Frame borders may be not match with the packet's borders. Thus after transmission via physical channel of part belongs to one packet, the part of other packet (transmitted via other virtual channel) will be transmitted. Thus transmission of rare part belongs to the first packet may be essentially delayed.

The mechanism of virtual channels allows to rationale divide physical channel's bandwidths between virtual channels. The bandwidth of a single virtual channel may be used inefficiently.

The mechanism of virtual channels allow to rationale division of physical channel's bandwidths between virtual channels. The bandwidth of one virtual channel may be used no effectively.

III. ON-BOARD NETWORK FOR VIDEO DATA STREAMING

Let's consider the part of onboard network represented on Fig. 1 as the use case.

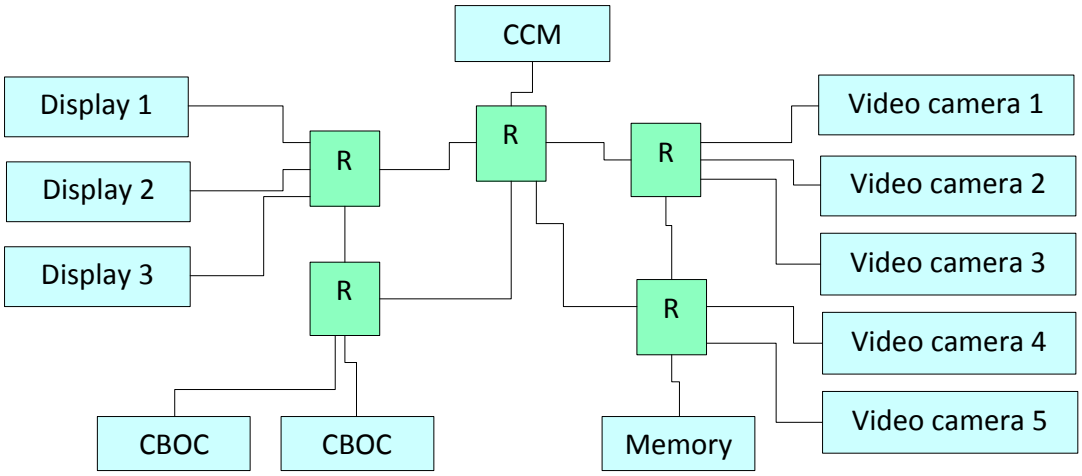


Fig. 1. An example of the part of onboard network

The types of traffic are transmitted via this network:

- Command traffic (packets with constrained size – 64 Bytes, transmission rate is very low, delivery time is critical);
- Real time video traffic (size of video frames could be about some Mbytes, they may be transmitted by one or some packets, data rate is high, delivery time and jitter of delivery time are important parameters)
- Other data – Best Effort (BE) traffic

In our use case data paths of these traffics are competing. All considered traffic types may be transmitted via one data link. Let's evaluate influence between the traffic types. It essentially depends on virtual channels parameters; therefore we make some decisions about these parameters.

We plan to use separate virtual channels for every traffic type. There are several sources of same traffic type (for example, several video cameras). Separate virtual channels could be used for traffic from every source or traffic from all sources could be transmitted via one virtual channel. (The quantity of virtual channels in the SpaceFibre is constrained by 32, but concrete device may support much less channels due to hardware constraints.)

A. Command traffic

The command traffic is most critical for the onboard network. Therefore we assign the highest priority to a virtual channel hat is selected for this traffic. The command could be transmitted via fixed or random timing intervals. Thus, its transmission could be permitted in all timeslots. The reserved bandwidth for this virtual channel should correspond to the rate of command traffic, in our use case this rate is 1 – 3%. If in a certain time moment the rate of command traffic would be higher than the reserved bandwidth, the commands will be transmitted as this virtual channel has highest priority during 1 ms – 1s (implementation defined). In most cases duration of increasing of command traffic is essentially less than this interval.

Let's evaluate influence of other traffic to command traffic delivery time. In worst case a command will wait for transmission of one SpaceFibre frame in every data link. A virtual channel for Command traffic has highest priority, therefore frames from it could wait for transmission of only one frame independently from quantity of virtual channels and its traffic. If transmission rate in a physical channel is 1,25 GBit/s transmission time of a frame with maximal length is 264 ns.

The graphs of dependency between command delivery time and quantity of transit routers are represented in Fig. 2. The graph «without other data» corresponds to the case when no other data is transmitted via the network; the SpaceFibre frame with a command will be translated via all data links without delays. The graph «with other data» corresponds to the case when the command waits for transmission of one SpaceFibre frame in every data link. The graph «with other data, Broadcast, FCT, ACK» corresponds to the case when the command waits for transmission of one SpaceFibre frame, one

Broadcast, one ACK and one FCT in every data link. These graphs show that maximal delivery time grows essentially with increasing of routers quantity, but in all cases is less than 35 us.

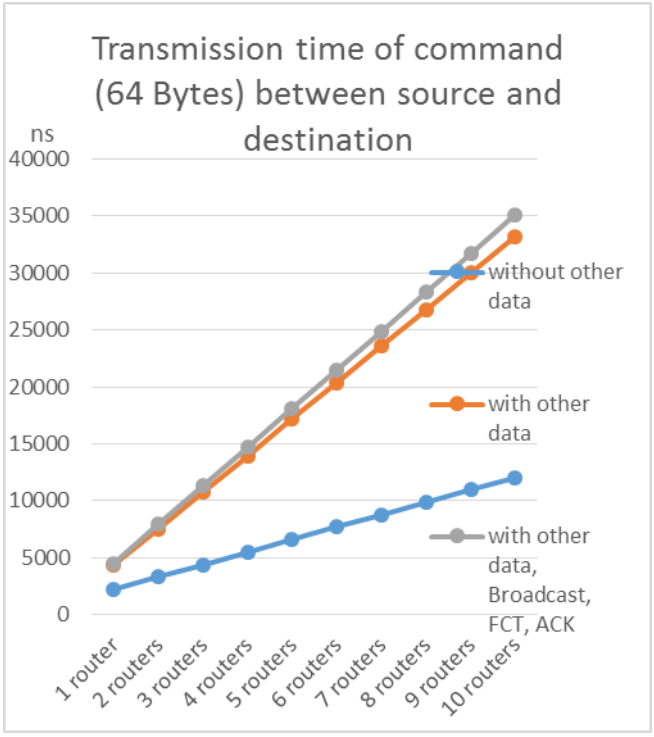


Fig. 2. The graphs of dependency between command delivery time and quantity of transit routers (transmission rate in the physical channel is 1,25 GBit/s)

B. Real time video traffic

The length of video frames is essentially bigger, than the length of commands. Transmission of video frames is periodic. If uncompressed video is used, the length of all video frames is equal, it can be 1 – 2 Mbytes and even more. If compressed video is used, the size of video frame depends from its type. The size of I-frame may be some Mbytes (typically a bit more than size of one uncompressed video frame). The size of P- and B- frames is about ten times less than the size of I-frames.

The maximal delivery time and jitter (jitter of delivery time) are critical parameters for real time video traffic. Therefore, we assign next level of priority after command traffic to a virtual channel that used for real time video transmission. The reserved bandwidth for this virtual channel depends on size of video frames and its rate. Video traffic is periodic, thus potentially we may use scheduling QoS for this traffic.

One video frame may be transmitted via the network by one or several packets at the transport/network layer. Correspondingly to the SpaceFibre standard, packet size does not lead to changes in delivery time for traffic that is transmitted via other virtual channels. Data are transmitted by frames with constrained maximal length at the data link layer. Selection of next virtual channel for transmission is implemented after transmission of every frame. Thus data from virtual channel with a high priority will be delayed no longer than transmission time of one SpaceFibre frame of maximal size.

Let's consider transmission of uncompressed video with the size of video frame 1 – 2Mbytes and the rate of 24 frames per second via the network. Required bandwidth for transmission of video frames with 1 Mbytes size is 25%, for transmission of video frames with size 2 Mbytes is 50%, when the channel rate is 1,25Mbit/s.

The graphs in Fig. 3 represents transmission time of one video frame via the path that includes one router and tree routers when other traffic does not transmit via the network (video data transmitted continuously).

These graphs shown that transmission time strongly depends from size of a video frame and practically does not depend from quantity of transit routers – delay of a SpaceFibre frame in one router is essentially less than delay of frame transmission via physical channel.

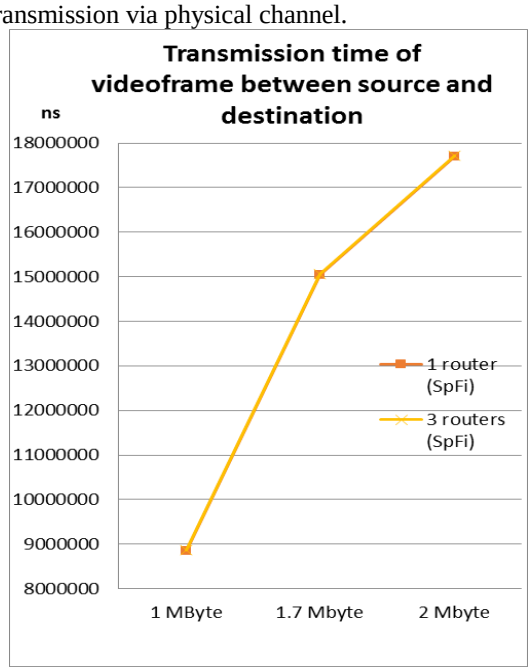


Fig. 3. The graphs of dependency between video frame transmission time and video frame size (channel rate 1,25Mbit|c)

When video flow transmission is realized by whole video frames a sequence of SpaceFibre frames, which belongs to one video frame, is transmitted to the network continuously (including the case when some transport/network packets are used for transmission of one frame). As result the SpaceFbre frames will be transmitted via every data link during long time (about 9 ms for 1 Mbyte length, about 18 ms for 2 Mbytes length). (Then data will not be transmitted via this virtual channel during long time.)

The width of Bandwidth credit counters in the network equipment should be enough for count during 9 ms (18 ms) without achieving Minimum Bandwidth credit Threshold.

If Minimum Bandwidth credit Threshold will be achieved the lowest priority level will be assigned automatically to this

virtual channel. In this case the video frame delivery time may be grow dramatically if other traffic is transmitted via the network.

Let's consider influence of other types of traffic to streaming (video) traffic. In our sample rate of command traffic is very low, packet length is 64 Bytes << max frame length. Therefore increasing of delivery time for video traffic is less than 1%.

Traffic with lower priority (for example, Best Effort traffic) may be transmitted via the network together with streaming traffic (compete with streaming traffic in routers). Therefore in the worst case every SpaceFibre frame, which belongs to the streaming traffic, will wait for transmission of one frame from BE traffic in every data link. It leads to increasing of streaming PDU's (video frame) delivery time in two times. If BE traffic may appear and disappear in different time moments, jitter of delivery time for streaming traffic will be about 9 ms (for PDU length 1 MByte), and about 18 ms (for PDU length 2 MByte).

There are some flows of video traffic between different sources and destinations in the network, which compete in routers. Let's suppose that all these flows are transmitted via one virtual channel (let's mark it VCi). If transmission of a packet from one source (let's mark it packet_1) to the output port of the router has started, the packet from other source (let's mark it packet_2) would not be transmitted to the same virtual channel (VCi) of this output port. The packet_2 will be transmitted to the output port only after transmission of packet_1 is finished. The next portion of data, which belongs to the packet_1, may be not ready (e.g. corresponding SpaceFibre frame may be delayed in a previous router), therefore the VCi in this case can stay idle during long time.

The packet_2 should wait of the virtual channel releasing in the router all this time. Therefore, delivery time of the packet_2 is increased in some times. The waiting time is proportional to the packet size. Therefore if one virtual channel is used for some data flows, the packet size at the transport/network layer should be strongly constrained (and should be essentially less than the size of video frame). However it will lead to essential increasing of overheads due to packets headers (and correspondingly decreasing of useful throughput).

If there is possibility of using a separate virtual channel for every source of video traffic (if tis traffic is competed in the network), then any constraint to SpW packet length is not required. If video traffic from all these sources has same parameters, the equal settings may be assigned to all virtual channels used for this traffic.

Let's consider the part of the network where video traffic from tree sources is competing. Size of a video frame is 1 Mbytes, and rate is 24 video frames per second. The equal priority (next priority after command traffic) and equal portion of bandwidth/throughput (25%) is set for every virtual channel assigned to video traffic.

In the best case all sources will transmit video frames in different time periods and BE traffic will not present in the network in these periods. In such case the video frame delivery time will be about 9 ms. In the worst case all sources will transmit video frames in one time and BE traffic will be also transmitted in this time. In this case worst delivery time for every SpaceFibre frame, which belongs to video traffic may be represented as the sum of waiting time (transmission time of one SpaceFibre frame, a frame from BE traffic and two SpaceFibre frames that belongs to video traffic from other sources) and transmission time of the considered frame. In our case, delivery time of video frame in this case will be in four times bigger than in case of empty network; its value will be about 36 ms, and, correspondingly, jitter will be about 27 ms.

Thus if there are Lv video data flows with same parameters and BE data flows (its parameters do not play any role) in the network, and they compete, the maximal delivery time of video frame is in Lv times bigger than delivery time of such video frame in empty network (without any other data).

In general case if in the network there are Lh data flows with priority higher than considered, (Lv-1) data flows with same priority and some data flows with lower priority (its quantity in not important), the maximal delivery time of a SpaceFibre frame, which belongs to the considered data flow, may be evaluated by next formula:

$$Tv_{\max} = \sum_{i=1}^{Lh} Th_i + \sum_{v=1}^{Lv} Tv_i + Tb \qquad (1)$$

where Thi – transmission time of SpaceFibre frame of maximal length, which belongs to highest priority traffic,

Tvi – transmission time of a SpaceFibre frame of maximal length, which belongs to traffic with same priority as the considered;

Tb –transmission time of a SpaceFibre frame of maximal length, which belongs to the traffic with lowest priority.

The maximal length of a SpaceFibre frame (and, correspondingly, its transmission time) for the concrete traffic in the network can be less than maximal length in the standard (256 bytes). This situation takes place when the packet length for considered traffic type is less, than 256 Nchars, and the data flow rate is small. In this case every packet will be transmitted in a separate SpaceFibre frame.

Jitter may be evaluated by the formula:

$$Tvj = Tv_{\max} - Tv_{\min} = \sum_{i=1}^{Lh} Th_i - \sum_{i=1}^{Lv} Tv_i + Tb - Tv \quad (2)$$

These evaluations show essential jitter for streaming (video) data flow. The jitter is in some times bigger than minimal delivery time of video frame.

Let's consider using of SpaceFibre scheduling QoS for jitter decreasing. We analyze an approach, when the timeslots and epoch changes in all data links of the network synchronously. The duration of a timeslot corresponds to transmission of every possible SpW packet in the network between source and destination.

In our use case rate of video frames is 24 frames per second. Correspondingly, every data source should have 24 timeslots for data transmission of every second. The timing interval between generation of two sequential video frames is

about 41,7 ms. Transmission time of one video frame via the network (when does not transmitted any other data) is about 9 ms (1Kbytes length), about 18 ms (2Kbytes length). We consider the sample of network with tree sources of video with length of video frame 1Kbytes. We select duration of epoch equal to 41,7 ms, and divide it to four timeslots with duration about 10,4 ms. Tree timeslots we assign to virtual channels for video traffic. Forth timeslot is used for BE traffic. As noted above the command traffic may be transmitted in all timeslots (also separate timeslot for this traffic is not required). With these settings the video traffic from one source can compete in the network with one SpaceFibre frame belongs to command traffic (influence of this traffic is minimal), and with one SpaceFibre frame belongs to other video traffic or BE traffic (waiting time is not more than time of one SpaceFibre frame transmission time). As result jitter is less than 1 us.

However, the sources of streaming data with different PDU's length (for example, compressed video) may be exist in the network. Using of such approach can lead to ineffective using of channel throughput. The duration of timeslots should be corresponds to transmission time of PDU's with maximal length. For example, in case of compressed video duration of timeslot should be enough for transmission of I-frame. But P-frames and B-frames length can be less than I-frame length in ten times. Quantity of transmitted P- and B- frames in dozen times greater than quantity of I-frames. Therefore in most timeslots the channel will be not used (data will be not transmitted) about 90% of timeslot's duration.

If there are some sources of streaming traffic with different and aliquant period of PDU's generation in the network, the task of timeslot duration's selection and quantity of timeslots in epoch selection is nontrivial.

C. Best effort traffic

The lowest layer of priority can be assigned for virtual channel, selected for Best Effort traffic (BE), because typically does not exist any restrictions for transmission time of this traffic type. Rate of BE traffic in some cases may be known, but in other cases it may vary essentially during system operation (BE traffic may have periodic or aperiodic nature). The portion of bandwidth reminder from other virtual channels, may be assigned for the BE traffic virtual channel. Data transmission via this virtual channel will be possible when there is no data for transmission in other virtual channels with higher priorities. In our use case the BE traffic compete in the network with command traffic and video traffic, therefore the maximal delivery time for BE traffic is about 27 ms.

In general case, if there are Lh command flows, Lv data flows of streaming (video) traffic (with equal parameters) and the Best Effort data flow (its parameters do not play any role) competing in the network, the maximal of transmission wait time for the BE traffic can be evaluated by the formula:

$$Tb_{\max} = \sum_{i=1}^{Lh} Th_i + \sum_{i=1}^{Lv} Tv_i \qquad (3)$$

Where Tvi – transmission time of a SpaceFibre frame that belongs to command traffic;

Tvi – transmission time of a SpaceFibre frame that belongs to streaming (video) traffic (in the empty network)

IV. Conclusion

In the paper we show that quantity and packet sizes of low priority traffic does not affect maximal delivery time of command traffic (highest priority traffic). The maximal delivery time of command traffic depends only from the number of transit routers. When path of the command traffic includes ten routers the maximal delivery time is less than 35 us; it is acceptable for most systems.

The evaluations of delivery time and jitter for streaming traffic (for example, video traffic) with real time requirements, when other traffic is transmitted via network, are represented in our paper. We show that when streaming (video) traffic from some sources is competed in the network it is appropriate to transmit the traffic from different sources via different virtual channels. If it is impossible (quantity of virtual channels implemented in the network equipment is not enough), the video traffic should be transmitted by small size packets (on the transport and network layer). However using of small size packets leads to increasing of transmission overheads, increasing of delivery time and decreasing of useful throughput.

We evaluate achievable timing parameters for streaming (video) traffic for considered use case when guaranteed bandwidth QoS is used (a separate virtual channel is used for every video traffic flow). The maximal delivery time of the video frame is less than 36 ms and jitter is less than 27 ms. These values do not depend on Best Effort traffic parameters in the network.

In this paper we show possibilities and constraints of using QoS scheduling when timeslot duration allows transmission of whole a PDU (for example whole video frame) between source and destination. The epochs and timeslots change synchronously in all data links of the network.

We show that this type of QoS allows to essentially decrease jitter for streaming traffic with equal sizes of PDUs (for example video traffic without compression). In our use case jitter is less than 1 us (jitter is equal 27 ms without scheduling). But this approach leads to essential decreasing of useful throughput when PDUs with different sizes are transmitted (for example compressed video). If there are some sources of streaming traffic with different and aliquant period of PDU's generation in the network, the selection of timeslot duration and quantity of timeslots in the epoch is nontrivial task for developer.

We show dependency between maximal transmission wait time for the Best Effort traffic and parameters of other traffic in the network. We evaluate maximal wait of transmission time for considered use case; it is about 27 ms.

Acknowledgement

The research leading to these results has received financial support from the Ministry of Education and Science of the

Russian Federation under grant agreement no. RFMEFI57814X0022.

REFERENCES

1 I. Korobkov “Adaptive Data Streaming Service for Onboard Spacecraft Networks”, Proceedings of 17th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT), 2015, pp. 291-298.

2 CCSDS 766.1-B-2 Digital Motion Imagery Recommended Standard

Test & Verification (Long)

How to design, test and verify the physical layer of SpW networks

SpaceWire Test and Verification session, Long Paper

Giorgio Magistrati *(Author)*, Norbert Bonnici, Wahida Gasti, Farid Guettache, Jorgen Ilstad
Data Systems Division ESA ESTEC
Noordwijk, The Netherlands
Giorgio.Magistrati@esa.int, Norbert.Bonnici@esa.int, Wahida.Gasti,@esa.int, Farid.Guettache@esa.int, Jorgen.Ilstad@esa.int

James Windsor, Science Directorate
ESA ESTEC
Noordwijk, The Netherlands
james.windsor@esa.int
Alex Palacios, Earth Observation directorate
ESA ESTEC
Noordwijk, The Netherlands
alex.palacios@esa.int

Abstract

The AIT/AIV spacecraft test campaign undergoes an electrical verification program for all the units and instruments. The SpaceWire standard calls for LVDS signals in the physical layer and these are considered critical because the present LVDS technology is capable to drive signals up to 400 Mbit/s with low common mode voltage. Reliable SpaceWire communication can be difficult in the presence of induced noise, ground level differences, impedance mismatches, failure to effectively bias for idle line conditions, and other hazards associated with installation of a SpW network within the spacecraft. Therefore, the verification steps of SpW signals are very important and they shall be addressed with special care.

Moreover, the new revision of the SpaceWire standard ECSS-E-ST-50-12C* rev.1 (currently under review) defines new requirements for the SpW signals in particular for the skew and the jitter budget and the margins. In addition, responsibilities for the two main actors (i.e. the unit manufacturer and the cable assembly manufacturer) are defined via jitter and skew budget calculation. All these aspects motivated ESA to the internal development of boards to perform the new requirement tests. This paper will summarize the related test campaign and the test coverage of various clauses of the new revision of the standard. A comprehensive plan and recommendations for the validation measurement of SpW link from the design to the final qualification/acceptance is proposed, distinguishing responsibilities among system architect, component manufacturer, unit supplier, cable assembly manufacturer and system integrator.

References to on-going ESA projects will be presented.. The effect on the overall skew and jitter budget calculation is discussed as well.

***Index Terms*—SpaceWire, Networking, Spacecraft Electronics.*)**

I. INTRODUCTION

After a mission is adopted, the related project planning and implementation encompass all of the processes carried out in order to plan and to execute a space project from initiation to completion at all levels in the customer-supplier chain in a coordinated, efficient and structured manner. Project breakdown structures provide the basis for creating a common understanding between all actors. They break the project down into basically 3 tree structures that are: the function tree, the specification tree and the product tree. The primary project product is the spacecraft. Following [1], the spacecraft product development is closely linked to specification, design, manufacturing and test activities on all the levels related to the spacecraft product tree supported by a V-Model process involving a succession of decisions based on reviews.

The V-Model process is thus designed as guidance for planning and executing the spacecraft product development taking into account the entire system (i.e. on-ground and on-board system products) life cycle development. It defines the results to be achieved and it describes the actual approaches for developing these results. At all levels of the product tree, the acceptance decision is linked to test results. In addition the V-Model specifies the responsibilities of each participant involved in each level. Thus, it is describing in detail, "who" has to do, "what" has to be done and "when" it has to be done within a spacecraft project.

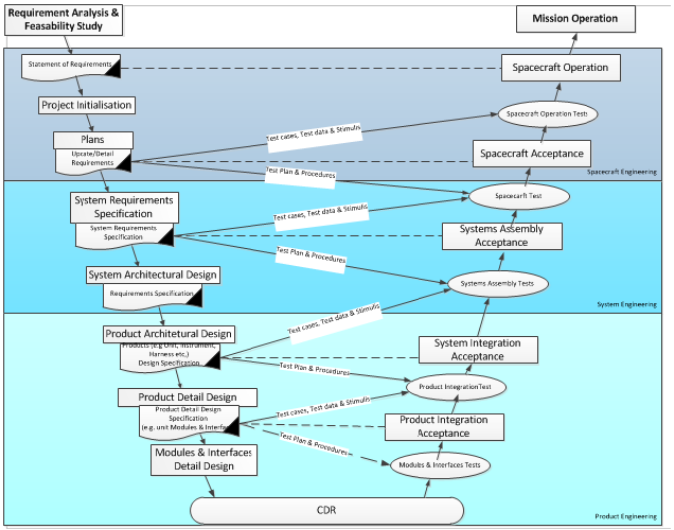


Figure-1: Basic structure of the V-model

Figure 1 indicates the basic structure of the V-model. It shows a top-down approach starting from the system specification down to the detail design of products at the lowest level. The first top-down phases are concluded by the Critical Design Review which should initiate the bottom-up approach. The bottom-up approach is punctuated with tests ranging from lowest level products tests (e.g. the units, the instruments, the harness etc.) to the final integration/assembly spacecraft acceptance tests.

Thus specifying the appropriate test cases for each requirement is essential to the V-model. The top-down structure of the requirement specifications and the design corresponds to a bottom-up structure of the requirement test cases.

The introduction of the V-model as a paradigm in the engineering flow of a spacecraft system results in a significantly more structured way of development within the supplier chain. The more the requirements are specified in detail, the more obvious the limited test coverage of complex assistance systems becomes.

In this paper, we present the specific case of how to design, to test and to verify the physical layer of SpW networks implemented within a spacecraft. The spacecraft SpW network specification at system level is based on the ECSS-E-50-12C. The paper will not address the SpW physical layer implementation over a backplane. The SpW standard covers three (physical, data-link and network) of the seven layers of the OSI model (ISO/IEC 7498-1) for communications. The ECSS-E-50-12C physical layer requirements are considering a point-to-point serial Data Link Interface depicted by Figure 2. The SpW Data Link Interface is constituted by a pair of LVDS driver/receiver exchanging signals via PCB tracks, connectors and an assembled cable with connectors as depicted by Figure 2.

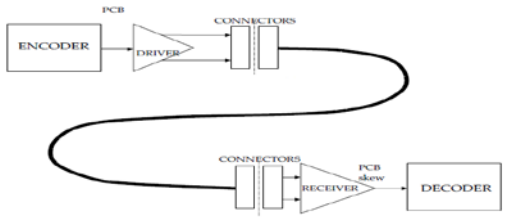


Figure 2: SpW Data Link Interface

The ECSS-E-50-12C is a specification made applicable to the overall supplier chain involved in the SpW Data Link Interface procurement. This standard does indicate any system apportionment requirements, thus it does not take into account the V-Model process so it does not cover the "who" has to do "what" and "when" responsibility aspects.

This is an issue for the spacecraft AIT/AIV work when the SpW Data Link Interface constituents are provided by different suppliers (i.e. unit supplier, instrument supplier and harness supplier). These suppliers have designed only one constituent of the SpW Data Link Interface, e.g. the driver node, and accordingly they have tested it only at their level and with EGSE representative to their level. The test coverage is only partial w.r.t the one related to the spacecraft. It is not addressed by the SpW standard how these supplier tests can be complemented at system level to ensure a reliable data communication over the SpW link. .

The new version of the SpaceWire standard [3], still under revision, describes a more thorough set of measurements for the SpaceWire Physical Layer under section 5.3 but it is still missing the top-down V-Model specification and design, and the bottom-up test approach.

Thus, this paper reports on the test of a sub-set of requirements related to the physical layer to introduce a discussion on the physical layer requirements of the SpW Standard and its revision 1 w.r.t to the V-model spacecraft development process. To this end,

- section II indicates the selected set of requirements supporting the discussion
- section III provides the review of the revision of the SpW standard
- section IV describes the test bench developed by ESA for the review of the new Standard
- Section V provides the test results and the preliminary comments
- The paper's conclusion (section VI) provides recommendations on how to design, test and verify the physical layer of SpW networks

II. SpW Physical Layer selected Requirements

In this section parts of the physical layer requirements presented in the new SpW standard currently under revision. The SpW Data Link interface requirements are falling under mainly three categories that are design requirements, signaling requirements and timing requirements. The signaling and timing requirements are linked to performance aspects that are mainly affected by the spacecraft grounding aspects (signaling) and with the spacecraft SpW network data rate aspects (timing). They are the two categories of requirements that have different test perspectives and constraints when addressed at unit or instrument level and when addressed at spacecraft level. The following Sub-sections are indicating the related set of requirements extracted from [3] for test and review.

A. SpW Physical Layer Signaling Requirements

The SpW signaling requirements are specified in section 5.3.5. Tables 1 and 2 contain the selected sets of requirements for which a test bench has been implemented and measurement test procedures have been established to ensure that a unit or an instrument is compliant with the standard.

The requirements are categorized in two parts:

1. The LVDS transmit signals requirements specified in section 5.3.5.2.2 (see Table 1)

Req.ID	Requirement description
a	When **terminated**, for measurement purposes, **by two 50 Ω ± 1%** termination resistors in series forming the required 100 Ω ± 1% termination impedance, the two outputs of the LVDS line driver (Out+ and Out-) shall have a **common mode voltage, Vcm**, measured at the junction of the two 50 Ω ± 1% termination resistors, of **1.125 V to 1.45 V**
b	When **terminated**, for measurement purposes, by a **100 ± 1 Ω termination** resistor, the two outputs of the LVDS line driver (Out+ and Out-) shall have amplitude, **Vtx, of +250 mV to +450 mV**, as illustrated in Figure 5-8, and a **differential amplitude** (Out+ minus Out-) of **2Vtx**
c	When a **logic 1** is to be transmitted **+Vtx shall be greater than –Vtx**

d	When a **logic 0** is to be transmitted **+Vtx shall be less than –Vtx**
e	The **steady state difference** in magnitude of the **common mode voltage, Vcm**, when transmitting a logic 1 compared to when transmitting a logic 0 shall be **less than 50 mV**.
f	The **steady state difference** in magnitude of **+Vtx or –Vtx** when transmitting logic 1 compared to when transmitting logic 0 shall be **less than 50 mV**.
g	The differential output of the line driver, Out+ - Out-, shall rise and fall monotonically with a **rise time** (Tr) and fall time (Tf) of **at least 260 ps** and **less than 0.3 times the bit period** (T), as illustrated in Figure 5-9, with the rise time being from **20% to 80%** of the difference between the two steady state values of the line driver differential output and the fall time being from 80% to 20% of those values.
h	The **differential output** of the line driver, Out+ - Out-, should **rise and fall** monotonically with a rise time (Tr) and fall time (Tf) of **less than 3 ns**.
i	**Ringing** on the differential output of the line driver, Out+ - Out-, shall **not be greater than ±0.4Vtx.**
j	The **maximum dynamic difference** in magnitude between +Vtx or –Vtx shall be **less than 150 mV.**

TABLE 1: LVDS transmit signal requirements

2. The LVDS receive signals requirement specified in section 5.3.5.2.3 (see Table 2)

Req.ID	Requirement description
a	The receive signals shall be terminated by a **100 ± 10 Ω termination resistor**.
b	The receive signals should be terminated by a **100 ± 1 Ω** termination resistor when an **external termination** resistor is being used.
c	The SpaceWire LVDS line receiver input **characteristics and sensitivity** shall be as defined in **TIA-644-A**.
d	A **differential signal greater than +100 mV** (i.e. +Vrx is greater than –Vrx by more than 100 mV) shall result in **logic 1** at the line receiver output.
e	A **differential signal less than -100 mV** (i.e. +Vrx is less than –Vrx by more than 100 mV) shall result in **logic 0** at the line receiver output.
f	The line receiver **shall maintain correct operation** for differential input voltages of **up to 600 mV** magnitude.
g	The line receiver shall **tolerate** a voltage on the **receiver inputs** in the range **0 V to +2.4 V** relative to the line receiver ground and operate correctly.
h	The line receiver **should tolerate** a voltage on the line **receiver inputs** beyond the range **0 V to +2.4 V** relative to the line receiver ground and operate correctly.

TABLE 2: LVDS receive signal requirements

B. The SpW Physical Layer Timing Requirements

The new SpW standard rev.1 [3] introduces an informative Annex A for the measurement of Data-Strobe Skew and Jitter. The requirements of Annex A contain the selected set of requirements for which a test bench has been implemented and measurement test procedures have been established to ensure that a unit or an instrument is compliant with the standard.

III. REVIEW OF THE NEW SPW STANDARD

A SpW communication channel implemented using a LVDS physical layer is depicted in fig. 5-7 of the standard and here below (see Figure 3):

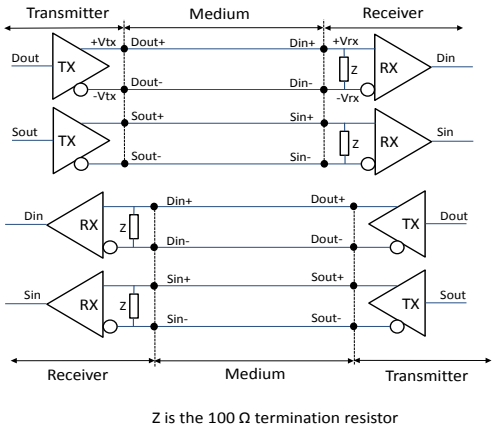


Figure 3: SpW Communication channel

However the figure presented in the Annex A of the new standard is considered more detailed: all the elements including source logic, PCB, connectors, cable assembly and destination logic that play an important role, are shown as per Figure 4.

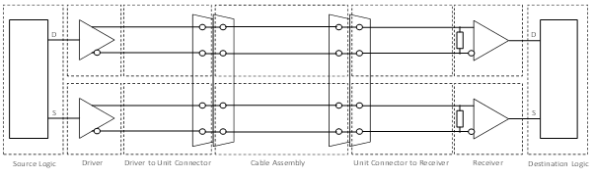


Figure 4: Elements of a SpW Communication channel (Sending node transmitting to receiving node from left to right)

Starting from Fig.4 we can add information related to the various stakeholders that are namely: Equipment Suppliers, harness (i.e. Cable Assembly) Manufacturer and Prime. The Prime role usually performed by two different people or entities: the System Architect that at the beginning of the project has to take care of the correct definition and specification of the various elements and the System Integrator that has to perform the integration and the verification at integrated level of the various units (and harness).

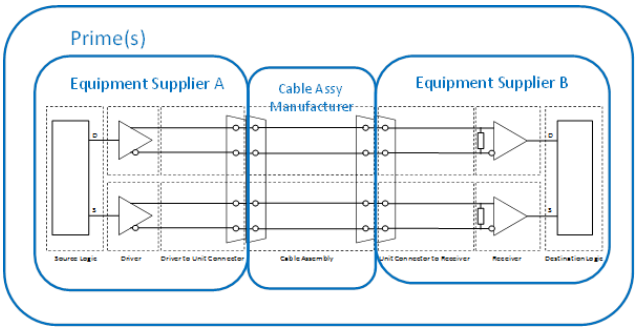


Figure 5: Elements and Responsibility of a SpW Communication channel (one direction)

Section 5.3.5.2.6 of the standard specifies requirements for the signal characteristic (edge separation) and quality (essentially differential amplitude) of the LVDS signal at the line receiver inputs. A test method is provided: repeated sequence of packets (composed by PSR stream and 0x55 pattern) to be measured across the line receiver termination resistor. The editor of the Standard is clearly aware that measurement at termination resistor level is not feasible, when the unit is closed and this is indeed the situation when an a model of an equipment is integrated on a test bench or on the Spacecraft for test, therefore clause 5.3.5.2.6e states: "*When access to the termination resistors is not possible, the receive signal may be measured at the connector adjusting for the transfer impedance between the connector and the termination resistor or other equivalent method*". This is a quite crucial point and we have severe doubts related to the feasibility of the first proposed method: the transfer function of the receiver board has to be provided and validated by the Equipment supplier but according to our experience this transfer function is usually not calculated by an Equipment supplier, additionally the transfer function of the connector that is part of the overall receiver board transfer function is usually not provided by the connector manufacturer.

A. *SpW LVDS "Port Replicator"*

We propose to follow a different approach: a LVDS port replicator could be interposed between the cable assembly and the Equipment's SpW LVDS circuitry and as such emulates this. This LVDS port replicator is essentially a LVDS receiver + driver. The interposition of the LVDS port replicator will affect the overall propagation delay of the SpW communication channel and it will add as well an extra contribution to the overall skew and jitter budget but it will allow to measure accurately the characteristic of the incoming LVDS signal in the specific implementation mainly in term of amplitude (differential and absolute) and timing.

For what concerns the amplitude of the signal it is important to underline that not only the differential amplitude (min and max as defined by 5.3.5.2.6a2&3) has to be measured but also the min and the max absolute voltages (as defined by 5.3.5.2.5k clause "*A line receiver input should withstand without failing a direct connection to a voltage between -0,3 V and +3,9 V relative to the driver [to be read as receiver] ground reference*" that means that the incoming signal should stay within this limit). The LVDS port replicator has to be electrically connected to the signal ground of the Unit under Test (and this could be done using the pin 3 of the connector in case of AL assembly). The LVDS port replicator must contain active parts and the voltage supply should be provided by batteries or receiver unit power supplies

The insertion of this LVDS port replicator would allow as well to measure the difference in the common ground as specified by 5.3.5.2.4a : "*The maximum potential difference between the local ground at one end of a SpaceWire link and the local ground at the other end of that SpaceWire link shall be between -1 V and 1 V*". It is assumed that the verification of this requirement is falling under the responsibility of the Prime(s): the system Architect during the architectural definition has to verify this aspect by analysis and system integrator during the integration phase will perform the final verification by test.

B. *Integrating SpW Units on a test bench or on a spacecraft*

Even if the LVDS port replicator makes it possible at the time of integration, to verifythe quality of the incoming signals in the case of an ideal implementation , w/o the transfer function of the Receiver unit (from connector to the receiver termination resistor), the real quality of the signals at the input of the receiver component cannot be assessed of a closed unit. In absence of the transfer function an estimation of the degradation in amplitude (in dB) and frequency spectrum could be proposed and applied to the measurements performed on the LVDS port replicator: the obtained figures could be finally compared with the requirements defined in section 5.3.5.2.6 of the standard.

Alternatively and this is indeed what it has been proposed by the Prime of Solar Orbiter in the Electrical Integration Test Plan signal characteristics to be measured directly at connector level of the unit have been specified.

The proposed values are applicable to both the SpW Input (Data and Strobe IN) and the Output (Data and Strobe Output) signals. The measurements are proposed to be taken using an ad-hoc test box, an Integrated Test Box (ITB), that has to have a minimum impact on the signal quality. The values proposed by the Prime are assuming a degradation introduced by the circuit and the Prime estimate that this degradation will not be worse than 7.6dB (from 240 mV down to 100mV that is the minimum specified as differential amplitude by the ANSI TIA 644 standard and section 5.3.5.2.2 of the standard). This degradation has not been independently confirmed and should be taken as a working hypothesis. Ultimately the impact of the attenuation is clearly a function of the data rate sought operated at and the length of the transmission line i.e. lower data rate means attenuation of the high frequency components of the rising and falling edges is affecting less the LVDS receiver's ability discriminate between 0's and 1's.

For the MTG Project the prime produced a technical note that synthesizes the context of the MTG-I and MTG-S Spacewire links, the definition of the grounding rails dedicated to these links, and the testing approach at unit level and system level. The technical note defines the measurement to be done at unit connector level for SpW signals and in particular eye diagram are required to be performed while triggering on the Data and Strobe signal positive and negative edge at +100mV for rising edge and -100mV for falling edge and also at 0V level. The eye diagram are used to derive the minimum edge separation without taking into consideration the possible effect of the unit connector and PCB on signal degradation. The applicability of figures and values defined by ANSI-TIA-644-A at unit level created some Non Conformances: e.g. the measurements of the LVDS dynamic output voltage during EQM qualification tests showed a variation higher than the 150 mV specified at the LVDS circuit by the ANSI-TIA-644-A and several clarification sessions within the MTG team were needed to resolve the non-conformance.

For BepiColombo, System AIT performed electrical measurements on the SpW interface following mounting of a SpW unit on the spacecraft. These tests involved inserting a dedicated

passive test-adaptor on the UUT's SpW connector and performing single-ended measurements of each line against the unit's signal ground (via Pin 3). The disadvantage of this approach is that if a non-conformance is measured then it is already too late and possible damage might have been caused. In cases where additional precautions were needed, a SpaceWire EGSE was used in between the units and measurements were taken on the SpW EGSE interface to the unit under suspicion because it has been observed that a substantially disturbed SpW signal has been cleaned by the EGSE unit and does not propagate further along the SpW link. However this means that single-ended measurements are made against the EGSE's signal ground and not the UTT so these measurements cannot be used to close out the test, meaning the SpW recorder must eventually be removed and the measurements repeated in order to complete the integration procedure. Ideally, having a means to perform safe measurements against the UUT's input interface vs. signal ground combined with having a clear requirement on the limits of the differential voltage would have allowed more efficient integration testing of payloads and data handling units on the spacecraft.

A LVDS port replicator can indeed be used to verify the quality of the LVDS signals generated by the driver circuit. However, it has to be underlined that the requirements defined in section 5.3.5.2.2 of the standard (LVDS transmit) are not directly applicable to measurements performed at unit/board connector level because they are specified at driver level (they are indeed directly taken from the ANSI-TIA-644-A), therefore also in this case a transfer function including the PCB and connector has to be subtracted to compare the obtained measurements with the requirements at driver components or even better based on the consideration that all the drivers components are compliant with the ANSI-TIA-644-A standard (therefore no need to verify this…) it is more important to specify a max degradation (induced by the PCB circuit and by the connector of the driver unit) in amplitude and timing characteristic of the signal at connector level that can be tolerated at system level (and this was what has been done by the Prime in Solar Orbiter).

C. Data and Strobe signals : Skew & Jitter budget

Section 5.3.6 of the standard defines and specifies how to measure the Data and Strobe skew budget and how to evaluate the maximum achievable operating frequency (inverse of minimum bit time) of a SpaceWire link. A margin of 10% is applied to the calculated skew figure. A more detailed explanation is included in Annex A of the standard where the various elements composing a SpW communication channel are individually addressed:

- source logic;
- driver;
- connections from driver to unit connector;
- cable assembly;
- connections from unit connector to receiver;
- receiver and destination logic.

Responsibilities are assigned for the specification and the measurement of the individual contributions to the overall skew and jitter budget, examples are:

5.3.2.6.2a. *"The equipment manufacturer shall provide the specification for the worst case skew between the differential data and strobe output signals at the SpaceWire connector of a unit (DSskewOUT), including the effect of transmitter jitter"*

and

Annex A- A3a. *"The cable assembly manufacturer provides the worst case skew between the differential data and strobe signals"*

However the proposed principle is not fully in line with a V-model verification process where the System Architect is supposed to be in charge of the architectural definition of all the communication channels and he is supposed, at the beginning of the project, to define an apportionment of the skew and jitter budget among the various contributors and to specify individual figures for all the units and the cable assemblies. The System Architect is in charge of maintaining the skew and jitter budget along the project life. It's a task normally shared with the System integrator at the end of the AIT phase. The Equipment supplier is in charge of the selection of components (source/receive logic and LVDS transmitter) and specification of the board/unit design in order to be compliant with the skew and jitter figures defined by the system architect. A similar role is performed by the Cable Assembly manufacturer for the cables and connectors. To be noted that the selection of components has to be done taking into consideration radiation, temperature range and aging. |Indeed all the manufacturer datasheets and SMD drawings for qualified components are defining values in the full T range and after irradiation.

The development of the units and cable assemblies supported by design justification files and analyses, if approved by the Prime/agency, can then be followed by the manufacturing phase. Measurements are needed at unit level and at integration level to confirm that the specified requirements have been fulfilled. Section 5.3.6 and Annex A are defining how to measure the jitter and skew contribution and the system integrator has to reassess the overall skew and jitter budget, to be noted that the measurement specified in section 5.3.6 and Annex A are done at ambient T and before exposition to radiation that will happen in orbit therefore margins or correction factor have to be applied on the measured values in order to compare the real measured budget with the estimation done at the beginning of the project and evaluate possible out of compliance. Availability of measurements done at different model (EM, EQM, PFM/FM) will make possible early verification of the skew and jitter budget with a risk reduction.

To be noted that the table presented in annex A (table A-1) and table 5-7 of the new standard erroneously does not include the jitter contribution of the Receiver unit.

D. Bandwidth of measurement tools

The standard requires that the operator has to use an oscilloscope and differential probes which have bandwidths of at least 1,05 times the reciprocal of the signal rise time.

BW(BandWidth) = 1,05/(signal rise time).

A justification note states that the factor of 1,05 is a rule of thumb requiring the bandwidth of the oscilloscope and probe to include the third harmonic of the signal edge bandwidth. The signal edge bandwidth is given by 0,35/rise-time (the bandwidth of a signal is 0,35/rise-time). The fastest signal rise time for LVDS is specified in ANSI-TIA-644-A to be 260 ps, resulting in a minimum combined oscilloscope and probe 3-dB bandwidth of 4 GHz. Allowing some margin the standard gives a recommended combined oscilloscope and probe 3-dB bandwidth of 5 GHz (which is the figure recommended in TIA-644-A).

The specified rule of thumb gives a quite accurate representation of the signal to be measured. All fast edges have an "infinite" spectrum of frequency components. However, there is an inflection (or "knee") in the frequency spectrum of fast edges where frequency components higher than F_{knee} are insignificant in determining the shape of the signal. To calculate F_{knee}:

$F_{knee}=0.4/T_{risetime}$ (20-80 percent)

F_{knee}=0.5/ $T_{risetime}$ (10-90 percent)
See [5]

Now, the above equations only tell us what the "useful" frequency content is in the signal we want to measure. We need further to specify to what accuracy we need to reproduce the signal. Depending on the frequency response of the scope, which is either Gaussian or maximally flat response, the F_{knee} can be multiplied with different accuracy factors to determine the necessary bandwidth required for an oscilloscope.

To achieve 3% accuracy the oscilloscope bandwidth needs to be: Fbw= 1.9xF_{knee} (Gaussian response) or 1.4xF_{knee} (Max. flat amplitude response). In contrast to achieve 10% accuracy, which is quite sufficient in most cases, the result is: BW= 1.3*F_{knee} (Gaussian response) or 1.2*F_{knee} (Max flat amplitude response).These equations are appropriate as a guideline to determine the maximum frequency the oscilloscope (including probes) should support to reproduce the measured signal sufficiently.

To be underlined that the recommendation to use a 5GHz BW oscilloscope in the standard is present also in section 5.3.5.2.6. where “system test” at the receiver unit are described, in this case the filtering effect of the parasitic capacitance (PCB, cable) are slowing down the edges, therefore the 260ps as fastest edge is not in reality a realistic value.

In Fig 6-8 rise and fall time of SpW signal have been measured at the output of a SpW unit, probes with different BW (1.5 GHz and 20GHz) coupled with High BW DSO and the measurements are showing that timing measurements are essentially the same (appx 340 ps).

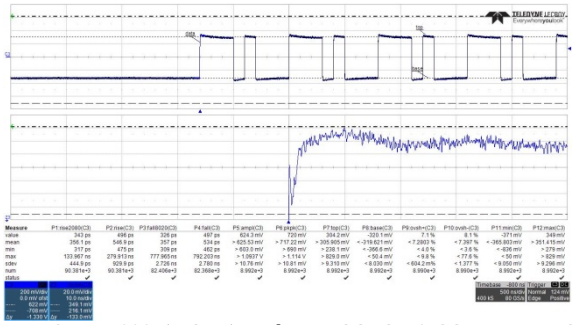


Figure 6: ZD1000 (1 GHz) probe - DSO @ 40 GSps – Data Signal (differential)

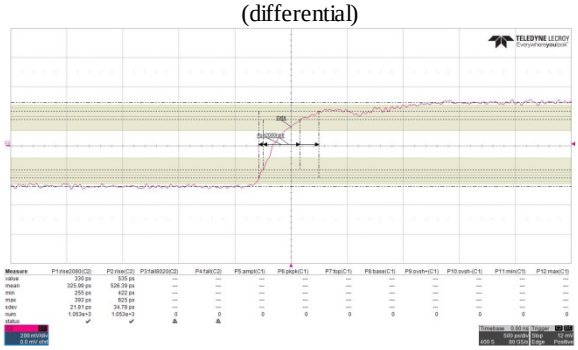


Figure 7: DS2005 (20 GHz) probe - DSO @ 80 GSps – Data Signal (differential) – rise time

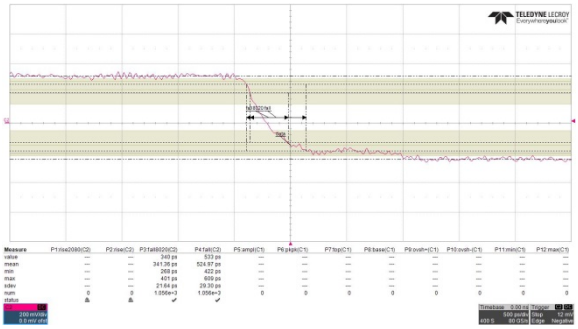


Figure 8: DS2005 (20 GHz) probe - DSO @ 80 GSps – Data Signal (differential) – fall time

IV. ESA Test Bench

A test bench has been defined and developed in order to test the set of requirements of the new SpW Standard defining the Physical layer of SpW:

a) Measurements at node level and at component level (to compare)

b) Measurements on Driver side
- Common mode (voltage, steady state difference, maximum dynamic difference)
- Differential mode (amplitude, steady state difference, rise and fall time, ringing)
- Jitter and skew contributions

c) Measurements on Receiver side
- Jitter and skew contributions
- Signal quality of received LVDS signal
- Absolute maximum ratings

Additionally significant effort has been spent in order to implement non-intrusive measurements techniques and to carefully place test point on the boards.

A.Test bench description

TEC-EDD has developed a board for testing and verification of SpaceWire communication channels (see Fig.9). The board hosts six (6) SpaceWire interfaces, four of them are nominal channels while two are redundant.

LVDS drivers and receivers from different manufacturers have been soldered (SpaceIC SPLVDS031/32 from SpaceIC, SN55LVDS31/32–SP from TI, and GR54LVDS049SPW LVDS dual-transceiver from Cobham Gaisler). Also two different types of Cross-Point Switches (SN55LVCP22-SP from TI, RHFLVDS2281 from ST) have been mounted and tested. SpaceWire codec IPs have been instantiated on an FPGA that hosts also the interface logic with the host computer.

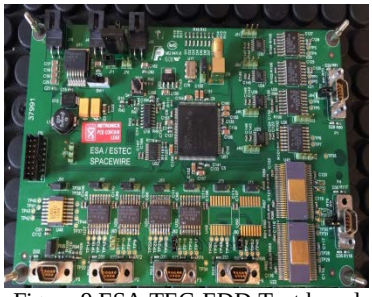


Figure 9 ESA TEC-EDD Test board

Several ad-hoc boards have been prototyped in order to perform the various measurement steps defined by the standard (Fig 10):

1. SpW interface mezzanine board (prototype)
2. SpW Unit Tester
3. SpW Driver Analyser
4. SpW Jitter Analyser

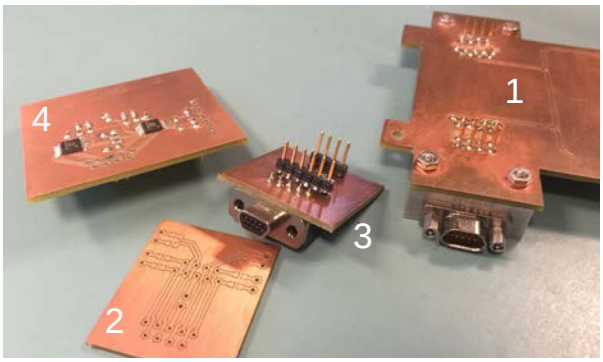


Figure 10 Test Jigs

The overall test bench is depicted by Figure 11.

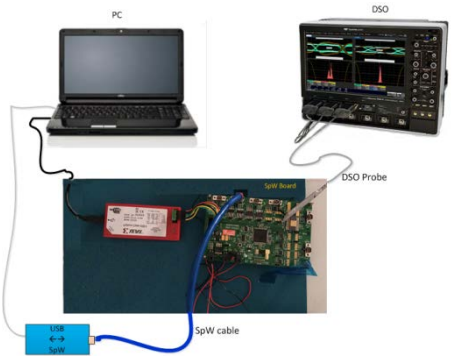


Figure 11: Test Bench Overview

V. Test results

Different techniques and probes with different BW (500MHz, 1GHz, 1.5GHz and 20GHz) coupled to DSO with high sampling rate and input BW have been used for the signals measurement.

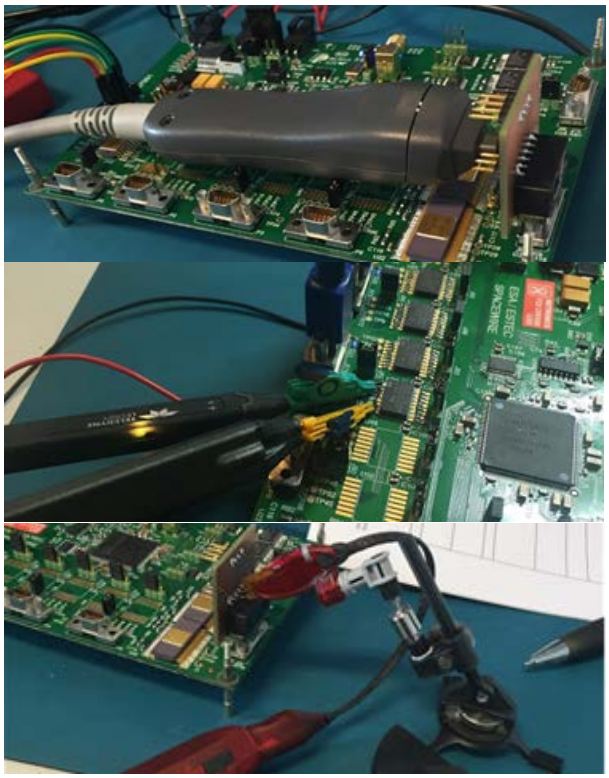


Figure 12 different Probes used on the ESA Test Bench

Several tests defined by the new SpW standard have been performed and hereafter the test results for the Eye diagrams measurements are reported: cables of 50 cm and 5 meters have been used and two type of patterns have been used (fixed pattern 0x55 and PSeudo Random-PSR pattern). The measurements have been done using or the built-in test point on the board or an external passive test jig (see fig.17)

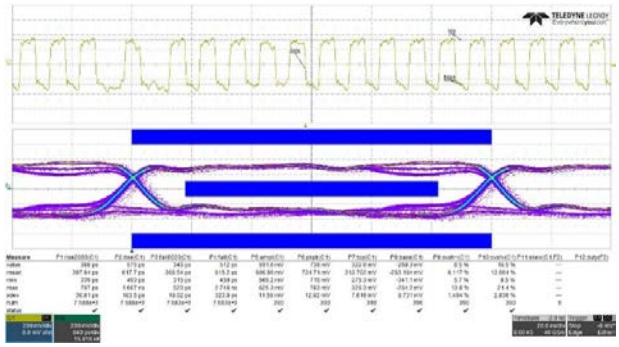


Fig 14 Eye diagram 1.5 GHz probe, 0x55 (>1000 bytes) using built-in test point - 5 mt cable, linkspeed is 100Mbit/s.

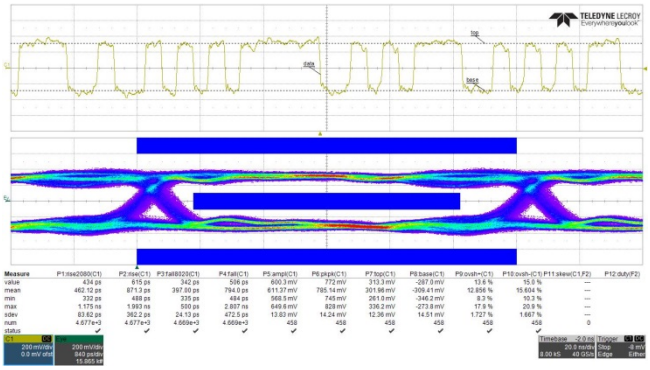


Fig. 15 Eye diagram 1.5 GHz probe, PSR (>1000 bytes) using built-in test point - 5 mt cable, link speed is 100Mbit/s.

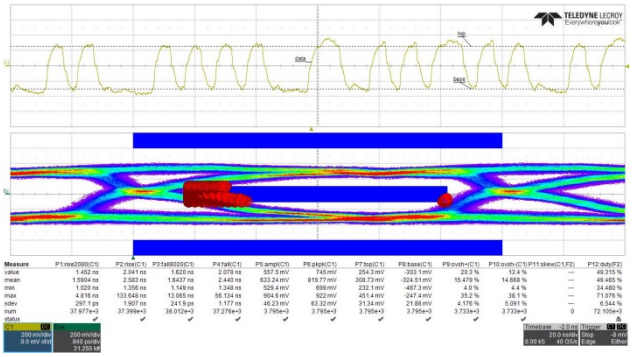


Fig. 16 Eye diagram 1.5 GHz probe –PSR (>1000 bytes) using external passive test-jig to measure the signal - 5 mt cable, linkspeed is 100Mbit/s

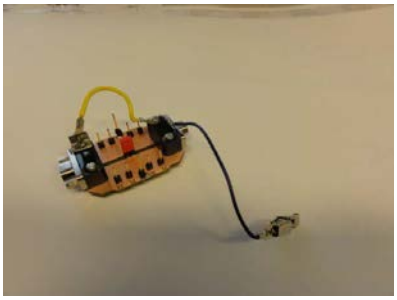


Fig. 17 External pssive test–jig used to measure SpW signal

Use of passive Test Jig or measurements at connector level of a unit can not be compared with ANSI-TIA/EIA-644-A limits unless corrected by impedance transfer functions that are particular for any given unit implementation. Firstly because it represents a considerable different load impedance than that for test fixture defined in the ANSI TIA/EIA-644-A standard. Alternatively margins have to be considered.

VI. Conclusion and Recommendations

The new SpW standard should assign precise responsibilities to the various actors involved in the design, development and verification of a SpW communication channel. The Prime as system architect has the responsibility to define and specify the performances of the communication channel at system and element level (unit, cable assembly). The element supplier has the task to provide a design that is compliant to the specified value and the compliance has to be demonstrated by analysis (that can refer also to datasheet). The verification process shall be started at unit and cable assembly level and it shall be again responsibility of the Prime to assess the overall compliance of the developed SpW communication channel to the system requirements.

Concerning the testing, it has been verified that in order to obtain the best results at board level it is necessary to place adequately the oscilloscope probes, e.g in the case of an eye diagram measurement at the LVDS receiver, the DSO probes should be preferably located on each side of the termination resistor. While for practical reasons it may not be feasible, any test point must be placed as close as possible to the termination resistor of each differential line. A test point should be small to reduce inductive as well as capacitive effects that distorts the signal, particularly the high frequency components of the rising and falling edges. Because of the fast edges of the LVDS signal low capacitance high bandwidth oscilloscope probes are required coupled with an oscilloscope with an adequate bandwidth, hence active probes that fulfill the previously mentioned rules for bandwidth calculation must be considered in order to give an accurate representation of the signal being measured (see [6], [8], [9]).

In case of unit-to-unit level tests the termination resistor within a unit may not be possible to access. In order to obtain the quality of the signal generated and received by the unit, an external adapter or port replicator, as reported in section III, could be introduced which presents the required differential impedance. It should as well present lowest possible distortion of the LVDS signal while allowing the SpW communication link between two end-points to operate at specified data rate. It is unfortunately not possible to achieve these overall goals using a passive adapter, particular because it does not support very well an undisturbed measurement between two end-points.

Although the circuit is relatively simple it presents some distortion to the LVDS signal mainly due to the additional MDM connector that is mated to the UUT. However the contribution in terms of distortion caused by the repeater board can be characterized using time domain reflectometer and subtracted from the measurements [7].

References

[1] ECSS-M-ST-10C. Project planning and Implementation, 6 March 2009

[2] ECSS-E-ST-50-12C. Space Wire - Links. Nodes. Routers and Networks.ECSS-E-ST-50-12V Rev.1, 1892, pp.68–73.

[3] ECSS-E-ST-50-12V Rev.1 May/November 2015 (DIR3).

[4] ANSI/TIA/EIA-644-A-2001. Electrical Characteristics of Low Voltage Differential Signaling (L VDS) Interface Circuits.

[5] Howard Johnson et.al. - A Handbook of Black Magic

[6] Understanding Data Eye Diagram Methodology for Analyzing High Speed Digital Signals, AND9075/D, ON Semi.

[7] Application note: TDR Impedance Measurements: A Foundation for Signal Integrity, Tektronix

[8] Application note: Probes and Probing, Teledyne Lecroy

[9] Application note: How oscilloscope Probes affect your measurement, Tektronix

A New Generation of SpaceWire Test and Development Equipment

SpaceWire Test and Verification, Long Paper

Stuart Mills, Chris McClements, Bruce Yu, Steve Parkes
STAR-Dundee
Dundee, Scotland
stuart.mills@star-dundee.com, chris.mcclements@star-dundee.com, bruce.yu@star-dundee.com, steve.parkes@star-dundee.com

***Abstract*—STAR-Dundee recently released a number of new SpaceWire test and development products based on a single hardware platform and supported by a single software platform. This paper will describe the modular design that makes this possible and the advantages, both to STAR-Dundee and to users, of this system.**

***Index Terms*—SpaceWire, SpaceFibre, STAR-Dundee, PXI, cPCI, PXIe, Interface, Router, STAR-System**

I. Introduction

STAR-Dundee has recently released a number of new SpaceWire test and development products based on a single hardware platform, using modular FPGA designs, and supported by a single software platform. The hardware, FPGA and software platforms each make use of a modular design, which allows different features to be included in a number of unique products.

This modular combination allows STAR-Dundee to quickly develop new products to support common requirements for SpaceWire and SpaceFibre test and development equipment. In addition, it provides a framework to explore new concepts without requiring completely new hardware, FPGA code and software to be developed.

This paper describes the hardware, FPGA and software modules which make up this system, and how they themselves have benefited from reusing previous developments. It then describes some of the products that have been released using this platform, and some of the projects that have used the platform to quickly develop devices to test out new technologies. The paper concludes with information on some new products being developed using the modules described.

II. Hardware Platform

To enable this modular system, a new hardware platform was developed. The STAR-Dundee PXI hardware platform has a CompactPCI (cPCI) connector at the rear. This allows the device to be used in cPCI, PXI and PXI Express (PXIe) racks. A photograph of the hardware platform is shown in Fig. 1.

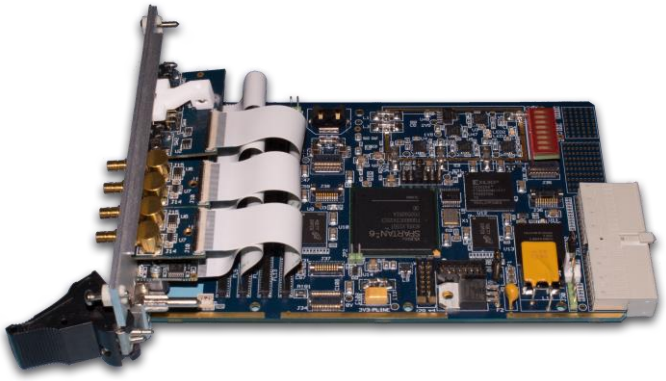


Fig. 1. STAR-Dundee PXI hardware platform

The hardware platform has been designed to support a number of different interfaces on the front panel, not only SpaceWire. The platform has sockets for sixteen flexi connectors, to which a number of different supported flexi interfaces can be connected and made available to users on the front panel of the device. The interfaces which can be connected currently include:

- SpaceWire ports
- SpaceFibre ports
- CAN bus ports
- JTAG ports
- USB UART ports
- GPIO ports
- SD card slots
- SMB trigger connectors
- Switches
- Push buttons

Other new interfaces can be developed and connected in the same way. Each interface includes LEDs which can be used to indicate status. For example, the SpaceWire interfaces have one LED which indicates whether packets are being transmitted, and another to indicate whether packets are being received. These LEDs can also be used to indicate errors.

Supporting each of these interface types allows devices to be created with a mixture of interfaces, for example a SpaceWire to SpaceFibre bridge or a device with multiple

SpaceWire ports and triggers, switches and buttons for triggering events.

Front panels must be manufactured to support the specified interfaces, but for internal developments or during prototyping these front panels can be quickly produced using a 3D printer. The PXI card is a 3U (rack Units) card, and the front panels are 3U high, with 6U versions available. The width of most front panels developed so far is 8HP (Horizontal Pitch), but larger or smaller widths such as 12HP or 4HP front panels can also be developed if required.

III. Reusable FPGA Modules

To support each of the interfaces which can be included in the hardware platform, FPGA modules have been developed for each interface. A number of these modules were developed for previous STAR-Dundee devices, or are modifications of existing STAR-Dundee FPGA modules.

Similarly, a module is required to interface with software over the cPCI interface. This is an existing module and provides the same interface as other STAR-Dundee devices such as the SpaceWire cPCI Mk2 [1], an older device with a cPCI interface. As well as minimising the FPGA development, this also reduces the software development required to support the PXI devices.

In addition to FPGA modules to support the device's interfaces, there are also FPGA modules to provide additional functionality within the device. For example, SpaceWire and SpaceFibre devices can include interface and/or router functionality. Other more advanced features that can be included are error injection on SpaceWire links, triggering on events and an RMAP (Remote Memory Access Protocol) target.

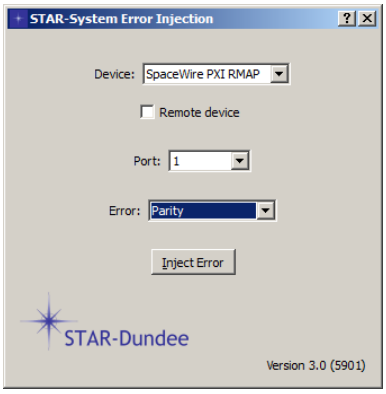


Fig. 2. STAR-System Error Injection application screenshot

IV. STAR-System Software

The STAR-Dundee STAR-System software suite [1][3] was developed prior to the PXI hardware platform. It provides a full software suite supporting all STAR-Dundee devices developed since 2012. At the bottom level it includes drivers for accessing each of the supported device types in the supported operating system. Above this are APIs for accessing these devices in software. At the top level STAR-System includes a number of console and Graphical User Interface (GUI) applications for accessing the devices. These include applications to transmit and receive packets and time-codes, configure the devices and inject errors. A screenshot of the STAR-System Error Injection application is shown in Fig. 2., while the Device Configuration application is shown in Fig. 3.

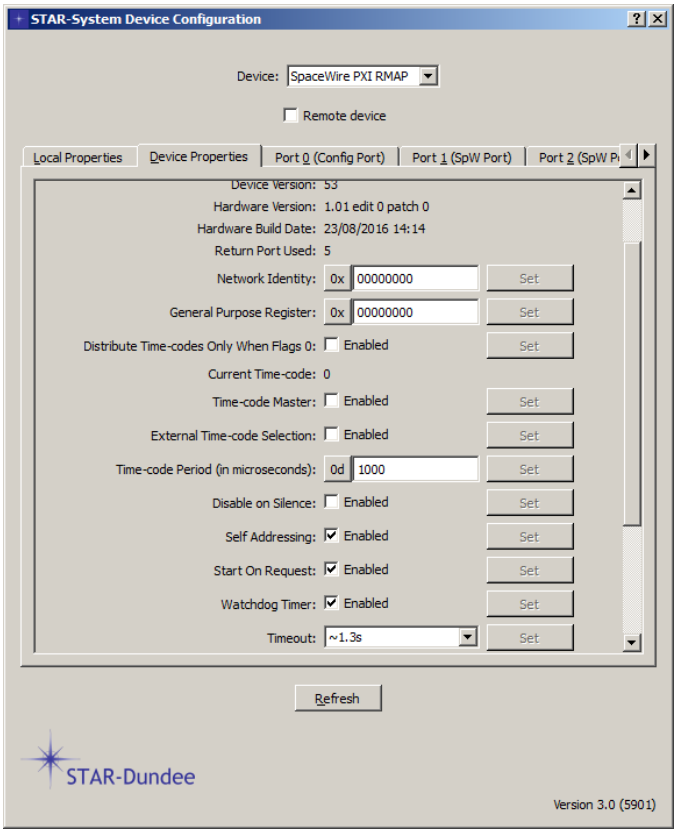


Fig. 3. STAR-System Device Configuration application screenshot

As the PXI devices use the same FPGA interface that is used in previous STAR-Dundee cPCI devices, the only modification required to the STAR-System drivers to support the PXI devices was to update the STAR-System PCI Driver to add support for the device identifiers used by each of the PXI products. Similarly the APIs were updated to include identifiers for each of the new products. No changes were required to the console and GUI applications, as these applications obtain device information from the APIs and drivers.

To support the unique features of the PXI devices, some additions were required to STAR-System. A new RMAP Target API was added to support PXI devices which contain an RMAP target. This API contains functions to configure the target, such as which commands are to be supported, and to receive notifications whenever an RMAP operation is performed. A Trigger API was also added to configure actions to be performed when specific events occur on devices supporting the triggering functionality. This is a powerful feature which can be used, for example, to transmit packets or time-codes when a particular event occurs, such as a time-code being received, an external trigger or a time period elapsing.

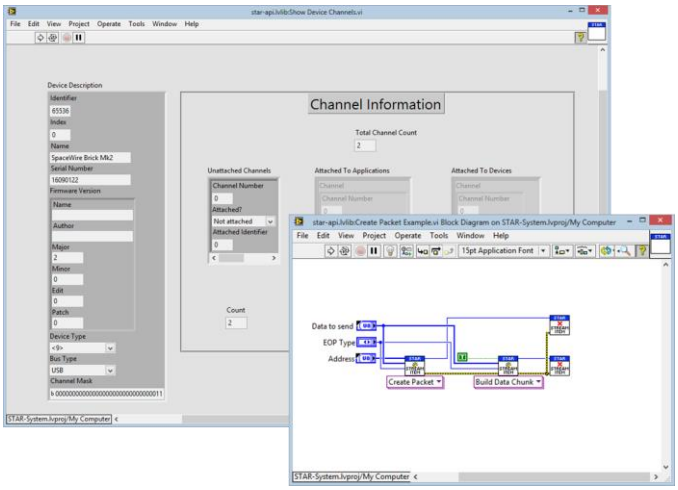


Fig. 4. STAR-System for LabVIEW screenshot

A. LabVIEW

Two options are available for using STAR-System devices with National Instruments' LabVIEW environment: a Windows LabVIEW Wrapper for STAR-System [4] and a LabVIEW VISA Driver [5]. A screenshot of the LabVIEW Wrapper is shown in Fig. 4. This Wrapper provides all the functionality of STAR-System within LabVIEW on Windows operating systems, while the VISA Driver offers lower level access to the device on any LabVIEW supported operating system.

As LabVIEW is often used on PXI systems, it was important to support the PXI products in both of STAR-Dundee's LabVIEW products. The VISA Driver required only minor modifications to support the additional device types. The LabVIEW Wrapper required similar modifications to support the additional device types, plus new modules to support each of the new APIs added to access the new functionality provided by some of the PXI devices.

V. PXI PRODUCTS

With the hardware, FPGA and software building blocks in place, these were then combined in to a number of different products, described below.

A. SpaceWire PXI Interface

The SpaceWire PXI Interface device [6], shown in Fig. 5., provides four SpaceWire ports, four SMB triggers, two push buttons and two switches on the front panel. The FPGA includes support for both interface and router modes, so the device can be used to explore SpaceWire routing, as well as transmitting and receiving directly on each of the four SpaceWire ports.

The inclusion of trigger connectors allows the device to make full use of the triggering functionality included in the FPGA, and supported in the STAR-System Trigger API. The Trigger API allows actions to be specified which will occur when specific events occur. The events include:

- An external trigger
- A valid time-code being received
- A counter being decremented or reaching zero
- An internal trigger
- A port event, including:
    - Receiving a start of packet
    - Receiving an End Of Packet (EOP)
    - Receiving an Error End of Packet (EEP)
    - Transmitting a start of packet
    - Transmitting an EOP
    - Packet available to be transmitted
    - Port running
    - Port encounters an error
    - Port disconnects
    - Port receives a time-code
    - Port transmits a time-code

The counter event can be used to delay a trigger for a specified period of time, or to trigger once a specified number of triggers have occurred.

The actions include:

- Output an external trigger
- Start or stop a counter
- Transmit a time-code
- A port action, including:
    - Transmit a queued packet
    - Disconnect the port
    - Inject a parity or escape error
    - Insert or suppress a Flow Control Token (FCT)
    - Increment or decrement credit

The combination of these actions and events allows very powerful control of the traffic on a SpaceWire link. There are numerous possibilities and users have been putting them to good use. One common use is to periodically transmit packets sent to the device from software, out of one or more SpaceWire ports. This provides deterministic behaviour while using a non-real-time operating system such as Windows.

Fig. 5. STAR-Dundee PXI Interface device

B. SpaceWire PXI Interface with RMAP Target

The SpaceWire PXI Interface with RMAP Target [6] demonstrates how a new product can be created with existing hardware. The device uses the same connectors and front panel as the SpaceWire PXI Interface, but in addition to the functionality provided with the SpaceWire PXI Interface, it also includes an FPGA module which provides four RMAP targets. An additional software API in STAR-System provides access to this functionality.

The RMAP Target module supports multiple targets, each of which can be configured to restrict the RMAP commands that are to be supported by that target. Authorisation of commands can be performed automatically by the device, or each command can be passed to software for authorisation. The properties that can be used when configuring automatic authorisation include:

- A logical address range
- A protocol ID
- Supported commands
- A key range
- A memory address range

A target can also be configured to notify software whenever a command is received and/or completed, while the memory on the device can be read or written from software. This provides a powerful system for testing of RMAP initiators and simulating RMAP targets, which can be setup very quickly.

C. SpaceWire PXI Router

The 16 flexi connectors on the STAR-Dundee PXI platform allow large SpaceWire routers to be created. The SpaceWire PXI Router [6] includes 12 SpaceWire ports, in order to fit all the ports in the 8HP front panel.

As with the PXI Interface devices, the SpaceWire PXI Router includes both interface and router modes, along with other features such as triggering support, although there are no external triggers on this device.

The SpaceWire PXI Router can therefore be used for similar purposes to the SpaceWire PXI Interface devices, while also offering the ability to explore and test SpaceWire routing with a large number of ports.

Fig. 6. STAR-Dundee PXI Router device

Fig. 7. STAR-Dundee SpaceWire Recorder

D. SpaceWire Recorder

The STAR-Dundee SpaceWire Recorder [7] is a rack system with a 1 Terabyte Solid-State Drive (SSD), which can record the SpaceWire traffic crossing up to four SpaceWire links. The large SSD allows the traffic crossing a network to be recorded for a much longer period of time than with a device such as the SpaceWire Link Analyser Mk2 [8], which makes use of internal memory for storage.

The SpaceWire Recorder rack, shown in Fig. 7., includes a SpaceWire Recorder PXI device to allow four SpaceWire ports to be monitored. This uses a front panel which is similar to the SpaceWire PXI Interface devices with four external triggers, two push buttons and two switches. The only difference is that the SpaceWire Recorder PXI includes eight SpaceWire ports.

The functionality provided by the FPGA of the SpaceWire Recorder PXI device is very different to that provided by the SpaceWire Interface and Router devices. It must transparently monitor the traffic passing between two ports, and provide this to software to be recorded. Some of the modules required to support this functionality were already available within STAR-Dundee's other products, however. For example, the SpaceWire Link Analyser Mk2 provides similar functionality, so some of this code was reused.

The software provided with the SpaceWire Recorder required much more development, however. Adding support for the device to STAR-System was a simple task, but the software to provide the functionality specific to the SpaceWire Recorder required considerably more development. This software must record the traffic to the SSD at very high speeds. It must then display the very large recordings to the user, working within the restrictions of the PC's limited memory.

Despite its unique nature, by using the SpaceWire PXI platform and other existing modules, the SpaceWire Recorder was developed in a very short time period, with a large percentage of that effort being focused on software development. The resulting product can be used to view the traffic crossing a SpaceWire network over large periods of time, quickly find errors, data patterns and time-codes, and can be an invaluable tool when debugging issues with a SpaceWire network.

VI. Projects Using PXI Devices

The PXI platform is of huge benefit for one-off developments, for example when developing devices for research projects. Devices can be quickly created using existing interfaces, or new interfaces can be developed in a relatively short period of time and added to the existing platform. Two projects which have benefited in this way are described below.

A. SpaceWire-D

As part of an ESA project on deterministic SpaceWire, the University of Dundee was required to produce a system demonstrating the capabilities of the SpaceWire-D protocol [9]. STAR-Dundee was given the task of developing a rack system with two routers, multiple RMAP targets, and two processors with SpaceWire interfaces acting as the RMAP initiators. The resulting SpaceWire-D Demonstration System is shown in Fig. 8.

In the Demonstration System, four SpaceWire PXI Interfaces with RMAP Targets are used to simulate as many as 16 RMAP targets. Two SpaceWire PXI Routers route traffic between the initiators and the targets.

The two RMAP initiators are provided by custom PXI devices. Unlike other STAR-Dundee PXI devices, these have a 12HP front panel. This allows nine SpaceWire ports to be included, three USB UARTs, four SMB triggers, two push buttons and two switches. The FPGA on these devices is also a custom development, which includes a LEON2 processor.

The University of Dundee was then able to use the RTEMS operating system on the initiators, and develop the SpaceWire-D initiator software to run on these boards. Software was also developed to run on the Windows operating system, using STAR-System and its RMAP Target API, to configure the routers and configure and monitor the RMAP targets.

The PXI hardware platform enabled this system to be developed in considerably less time than would have otherwise been possible, allowing University of Dundee to concentrate on the research and development of the SpaceWire-D software.

Fig. 8. SpaceWire-D Demonstration System

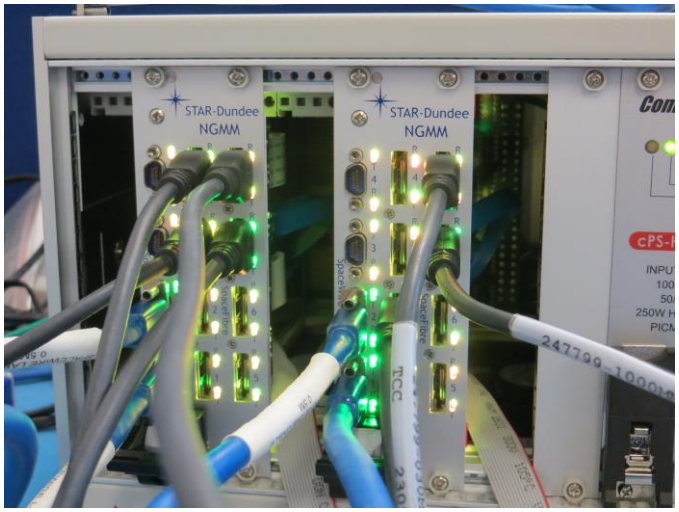


Fig. 9. SUNRISE router devices routing SpaceFibre and SpaceWire packets

B. SUNRISE

STAR-Dundee has been leading the development of SpaceFibre, and has been working on the development of a SpaceFibre Router under a Centre for Earth Observation Instrumentation and Space Technology (CEOI-ST) activity called SUNRISE.

The hardware for the SUNRISE SpaceFibre Routers is provided by the PXI platform. It makes use of the SpaceWire and SpaceFibre interfaces, including eight SpaceFibre ports and four SpaceWire ports. Two SUNRISE routers are shown in Fig. 9. routing traffic between SpaceWire and SpaceFibre ports.

The PXI hardware platform allowed devices to quickly be created so that work could instead focus on the core objective of the activity: developing the FPGA module to perform SpaceFibre routing. This has been a very successful project and resulted in the development of the first ever SpaceFibre router, enabled by the PXI platform.

VII. Conclusion

This paper has described the building blocks that make up the STAR-Dundee PXI products, and has shown how these hardware, FPGA and software modules can be used to produce a wide range of products while also providing a platform for prototyping and experimentation.

Work is continuing on this platform, and more products will be released, as a result of FPGA and software additions, with new front panels designed when required. Potential future products include a SpaceFibre interface device and a SpaceWire to SpaceFibre bridge.

There is work also being performed at STAR-Dundee in a slightly different direction, to take advantage of the existing interface boards, software and FPGA modules. A new hardware platform has been developed with a PXIe connector at the rear, but using the existing front panel interfaces. This platform includes a Microsemi RTG4 FPGA – a flash-based radiation tolerant FPGA. The PXIe-RTG4 platform, shown in Fig. 10., offers the same flexibility as the PXI platform, while

providing a high quality engineering prototype board for the development of RTG4 applications.

Fig. 10. STAR-Dundee PXIe-RTG4 device

The PXI hardware platform, reusable FPGA modules and STAR-System software suite provide a powerful combination which enables STAR-Dundee to develop bespoke products to meet customers' requirements. With the possibility of alternative hardware platforms, additional interfaces and new FPGA and software modules, the platform will continue to be developed as new requirements are identified which cannot be met with the existing system.

ACKNOWLEDGMENT

The STAR-Dundee PXI platform is the result of a combined effort between STAR-Dundee's PCB, mechanical, electronics, FPGA and software engineers. The authors would like to acknowledge the efforts and input of the STAR-Dundee engineers not listed as co-authors.

STAR-Dundee would also like to acknowledge the support of the CEOI-ST in funding the SUNRISE activity, Contract Number: RP10G0348A02.

REFERENCES

[1] STAR-Dundee, "SpaceWire cPCI Mk2", https://www.star-dundee.com/products/spacewire-cpci-mk2, 2016.

[2] S. Mills, A. Mason, C. McClements, D. Paterson, I. Martin, S. Parkes, "Developing SpaceWire Devices with STAR-Dundee Test and Development Equipment", International SpaceWire Conference 2013, Gothenburg, Sweden, June 2014.

[3] STAR-Dundee, "STAR-System Datasheet", https://www.star-dundee.com/sites/default/files/STAR-System_1.pdf, 2016.

[4] STAR-Dundee, "STAR-System for LabVIEW", https://www.star-dundee.com/products/star-system-labview, 2016.

[5] STAR-Dundee, "SpaceWire LabVIEW (VISA) Driver", https://www.star-dundee.com/products/spacewire-labview-visa-driver, 2016.

[6] STAR-Dundee, "SpaceWire PXI", https://www.star-dundee.com/products/spacewire-pxi, 2016.

[7] STAR-Dundee, "SpaceWire Recorder", https://www.star-dundee.com/products/spacewire-recorder, 2016.

[8] STAR-Dundee, "SpaceWire Link Analyser Mk2", https://www.star-dundee.com/products/spacewire-link-analyser-mk2, 2016.

[9] D. Gibson, S. Parkes, C. McClements, S. Mills, "SpaceWire-D Prototype and Demonstration System", International SpaceWire Conference 2016, Yokohama, Japan, October 2016.

Software-to-Hardware Tester for SpaceWire Oriented Transport Protocols

SpaceWire networks and protocols, Long Paper

Valentin Olenev, Irina Lavrovskaya, Nadezhda Chumakova
Saint-Petersburg State University of Aerospace Instrumentation
Saint Petersburg, Russia
{valentin.olenev, irina.lavrovskaya, nadezhda.chumakova}@guap.ru

Dmitry Dymov, Vadim Shkolniy, Sergey Kochura
JSC "Academician M.F. Reshetnev" Information Satellite Systems"
Zheleznogorsk, Russia
dymovdv@mail.ru, {shkolniy, kochura}@iss-reshetnev.ru

***Abstract*—Compliance or conformance testing is basically a kind of an audit which is performed for the system to check whether all the specified standards are met or not. Implementation of conformance testers for the communication protocols is an important task, which is being solved in the majority of industrial companies that develop the communication equipment.**

On-board equipment always needs a proper testing before the integration into a spacecraft. Especially if we talk about equipment, that operates according to the newly developed communication protocol. Conformance testing is such kind of testing, which gives an ability to ensure that a hardware or software product, system or a physical link complies with the requirements of a specification or any other document.

There is a number of transport protocols intended to operate over SpaceWire. The newly developed transport protocol STP-ISS is now among these protocols and provides such services as reliability, guaranteed data delivery, scheduling and connection-oriented data transmission. In order to test the software models or hardware implementation of the STP-ISS protocol the authors created a so-called software-to-hardware tester. It gives ability to test the real on-board hardware with the software implementation of a protocol model. The evolution of a tester can provide opportunities for testing other SpaceWire oriented transport protocols such as RMAP, CCSDS PTP, SpaceWire-R, etc.

The current paper gives an overview of conformance testers, describes main features of the STP-ISS protocol, and, finally deals with the implementation of the Software-to-Hardware STP-ISS tester and its application use cases.

***Index Terms*— Conformance Tester, SpaceWire, STP-ISS, Transport Protocol.**

I. Introduction

Nowadays, it is a common practice that industrial companies develop special testing equipment or software in order to ensure that standard compliances are met. The widely used standards such as USB or Ethernet are developed by large organizations and could be tested on all stages of the implementation. We do not need a special equipment to test the hardware if we buy a USB stick or a networking card, we can just plug it into a computer and operation system will do it automatically. But if a company develops a new specialized protocol and a number of devices that should meet the requirements of a new standard, this company should carefully test the implemented equipment before integration and dissemination.

On-board equipment is such kind of equipment that needs very proper and detailed testing [1]. And if a new protocol for on-board communication is developed, then we need to be sure that the devices work as expected, before we can integrate it into an aircraft or a spacecraft.

We had a long-term project for the research, development and implementation of an STP-ISS transport protocol for the on-board communication via the SpaceWire networks. In this project we developed two revisions of STP-ISS protocol, simulated and investigated them. The first revision of STP-ISS is much simpler and compact, but the second one is more powerful. Nevertheless, the backward compatibility for these revisions is provided. After that we got the task to implement a tester for the STP-ISS rev.1 equipment, which could tell the manufacturer, that STP-ISS device operates correctly. Tester should examine the device with a set of different testing scenarios; each scenario should test a particular STP-ISS mechanism. So after the testing the manufacturer will know what STP-ISS mechanism failed and it can analyse the log-files for details.

For this reason we conducted an overview of different approaches for the implementation of hardware conformance testers, studied the main examples of conformance testers represented at the market.

II. Hardware and Software Conformance Testing

On-board equipment always needs a proper testing before the integration into a spacecraft. Especially if we talk about

equipment, that operates according to the newly developed communication protocol. The conformance testing should be provided to prove that this equipment meets the requirements.

Conformance testing gives an ability to ensure that a hardware or software product, system or just a medium complies with the requirements of a specification or any other document. Various test procedures, testing software or hardware testers have been developed either by the standard's maintainers or external auditing organizations, specifically for testing conformance to standards. Also service providers, equipment manufacturers, and equipment suppliers rely on such testing to ensure Quality of Service through this conformance process.

Conformance testing may include some of these kinds of tests, it has one fundamental difference – the requirements or criteria for conformance must be specified in the standard or specification. This is usually in a conformance clause or conformance statement, but sometimes some of the criteria can be found in the body of the specification. Some standards have subsequent documentation for the test methodology and assertions to be tested. If the criteria or requirements for conformance are not specified, there can be no conformance testing [2].

Many companies that develop or just work with the new equipment have such kind of conformance testers and usually equipment testing is done by the testing organizations. But some standards have no official testing organizations. They rely on self-assessment by the implementer and acceptance testing by buyers.

Depending on the available information we can elaborate two main approaches for the conformance testing that are widely used across the industry:

- Software testers;
- Hardware testers.

Software testers usually consist of a test entity (software) that includes a number of test cases. These tests are aimed to get the correct responses from the unit that is being tested. Testing software is running on a PC or any portable device and it is connected with the real hardware, that it tests. Conformance testing software usually includes a test tool (e.g., tool, suite, and/or reference implementation) and procedures for testing (test engine).

The software may be represented by a set of programs, a set of instructions for manual action, or another appropriate alternative. It is likely to be platform independent, and it should generate repeatable results. A reference implementation is an implementation of a standard that is by definition conformant to that standard. Such an implementation provides a proof of concept of the standard and also provides a tool for the developers of the conformance software. The reference implementation is of considerable importance on the early stages of conformance testing.

The conformance testing procedures should be agreed and implemented before testing begins. This would include the implementation of different types of tests.

There are many examples of the software testers for the communication protocols. One of them is the "HDMI compliance test software" that is implemented and distributed by Tektronix. It automates a comprehensive range of tests on conformance to HDMI 1.4a/b and HDMI 2.0 standards (see Fig 1) [3].

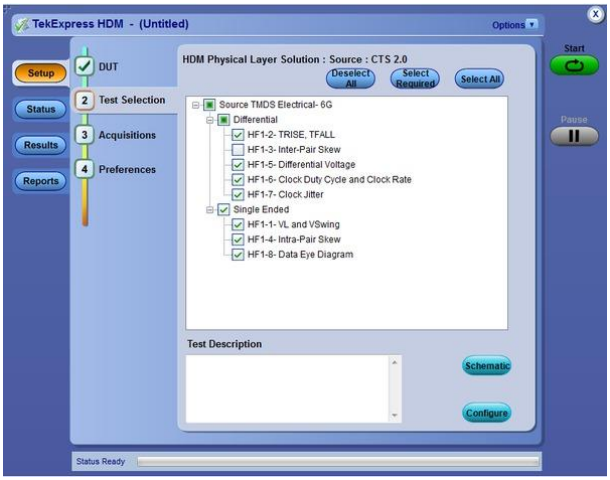


Fig. 1. HDMI compliance test software by Tektronix

The other good example of testing software is R&S®CMW – Conformance testing solution for eCall/ERA-Glonass implemented by Rohde&Schwartz Gmbh&Co (see Fig. 2). It is electronic safety systems for cars, developed by European Union and the Russian Federation to have intelligent telematics-based vehicle safety systems to speed up emergency response times in order to save human lives. This software tester is a solution for automated, reliable and reproducible end-to-end conformance tests on eCall/ERA-Glonass modules [4].

Fig. 2. Conformance testing solution for eCall/ERA-Glonass implemented by Rohde&Schwartz Gmbh&Co

Also there are a number of good conformance testing software implementations based on the formalized methodics and algorithms. These examples are described in [5] and [6].

The other way of conformance testing is using of the real hardware testers that produce the test sequences and test the remote device. Usually it is a device operating with full respect to the standard, which could have a different number of parameters and settings. Configuration of this device is performed via a special configuration software installed on PC.

There are also many examples of hardware testers for the widely used communication standards as USB, LAN, RS232 and others (see Fig. 3).

Fig. 3. Implementations of hardware testers for different communication standards

There is another example that is related to the on-board equipment testing – the SpaceWire Conformance Tester implemented by Star-Dundee. It connects to a SpaceWire device and, through the host software, executes a variety of tests to check the device under test's (DUT) compliance to the SpaceWire Standard. Over 55 tests can be conducted. The user can easily select which tests they do and do not want to run. With each test, expected and achieved results are displayed, including a link to the appropriate clause of the SpaceWire Standard to dramatically reduce the time spent debugging the DUT. The SpaceWire Conformance Tester can also be used as a high speed packet generator, and one of the SpaceWire links can act as a data / time-code sink or loop-back [7].

Fig. 4. SpaceWire Conformance Tester

III. STP-ISS Protocol Features

The main task for our research was to decide how to test the newly developed devices that should operate in conformance to the STP-ISS protocol specification. So firstly, we should describe what STP-ISS protocol is, what main mechanisms and distinctive features it has. In this section, we consider the second revision of the STP-ISS protocol which includes all required functionality.

STP-ISS is a transport layer protocol that describes informational and logic interaction between on-board devices, packets' formats and packet transmission rules for SpaceWire networks.

STP-ISS provides transmission of control commands, application messages, SpaceWire time-codes, SpaceWire distributed interrupts and interrupt-acknowledges. There are two types of application messages:

- urgent messages (higher priority);
- common messages (lower priority).

STP-ISS encapsulates applications' messages into SpaceWire packets. Length of each message data block should be not less than 1 byte and should not exceed 2048 bytes for the connectionless data transmission, and 64 Kbytes maximum for the connection-oriented data transmission. Each packet is finalized with CRC-16 which covers the packet starting from the first byte of the STP-ISS packet header (excepting path address) till the last byte of data, excluding the end of packet symbol EOP.

For each transmitted packet STP-ISS protocol has a special lifetime timer, which counts the time, when the packet is still relevant in the SpaceWire network. Each packet is stored in a transmission buffer during its lifetime.

STP-ISS has two logical buffers at the receiver side. The first buffer is used for the connectionless data transmission, for all types of packets (control commands, common messages and urgent messages). The second buffer is used for the connection-oriented data transmission only. The receiving side should reserve required space in the buffer for each new connection. If one of the receiving buffers is full, then STP-ISS should indicate the Application layer about it and discard all the packets coming from the SpaceWire.

The important STP-ISS feature is its configuration flexibility. The protocol has a number of configuration parameters, which give ability to tune the protocol depending on the developer needs. There are some mechanisms that should be implemented as mandatory. For example, Priority QoS at least for one priority, Best effort QoS, transmit and receive buffers. The other mechanisms are extensions and could be optionally implemented in different combinations.

One of the STP-ISS benefits is the possibility to transmit data using the following quality of service types:

- priority quality of service;
- guaranteed delivery quality of service;
- best effort quality of service;
- scheduling quality of service.

A. Priority Quality of Service

Priority quality of service is the main quality of service type that should be supported by all the network end-node devices, which communicate by means of STP-ISS. According to this quality of service type, the data with the higher priority should be transmitted first. STP-ISS supports 9 levels of priorities.

B. Guaranteed Delivery Quality of Service

Guaranteed delivery quality of service provides confirmation for the successful packet transmission by sending the acknowledgement packets. In addition, it resends the data from the transmitter end-node if the acknowledgement is lost

(resending mechanism). Guaranteed delivery is provided by resend timers and acknowledges.

Another feature is duplicate control commands detection in the receiver. A duplicate control command can occur in case of a loss of an acknowledgement.

C. Best Effort Quality of Service

Best effort quality of service provides data transmission without acknowledging. When an STP-ISS receiver gets a best effort packet it checks the CRC and data length only. In case of an error or if the packet ends with EEP, the data packet still should be sent to the Application, but with an error indication.

D. Scheduling Quality of Service

STP-ISS assumes to have a single data transmission schedule for the whole SpaceWire network. This schedule gives an opportunity for the node to send data only during particular time-slots. The schedule consists of a number of time-slots. The schedule table describes one epoch.

STP-ISS has the timer synchronisation mechanism. Synchronisation is performed once in an epoch. During synchronisation, a node should calculate a new value for the time-slot timer. The newly calculated value will be applied for the time-slot timer of a new epoch. The new epoch should start when the time-code is received.

There are *K* time-slots in each epoch, when the time-code is recognized as relevant. These time-slots are called Time-code relevancy window. If a time-code is received before the last *K/2* time-slots of the epoch, or after the first *K/2* time-slots of the epoch, then this time-code is considered as irrelevant and synchronisation should not be performed. If the time-slot timer for a last time-slot expires simultaneously with the time-code reception, then there is no need to correct the epoch timer value.

E. Connection-Oriented Data Transmission

Connection-oriented data transmission gives an ability to transmit large sized data with minimum overheads. Only urgent or common messages could be transmitted over a transport connection. Maximum number of transport connections should not be more than 8 per one direction. Each transport connection is unidirectional: it connects the transmitter of the initiator node and receiver of the remote node.

An application, which needs to transmit or receive a large portion of data, should initiate the transport connection establishment. The maximum size of data, which could be transmitted over the transport connection in a packet, is 64 Kbytes. The transport connection establishment is performed by means of classical three-phase handshake [9], [10].

During data transmission, STP-ISS provides the flow control, which is performed by sending of information about the available free space in the receiving buffer. This mechanism is applied only for the transport connections with the guaranteed quality of service.

STP-ISS rev.1 protocol is described in [11] while STP-ISS rev.2 protocol was previously described in details in [12].

IV. STP-ISS Reference Code

STP-ISS specification development was followed by a simulation phase [13]. During this simulation stage we precisely analysed, investigated and tested the specification. In order to check STP-ISS protocol mechanisms we used three different models:

- SDL model;
- SystemC network model;
- C++ reference code.

These modeling and investigation directions for STP-ISS were described in more details in [14].

The SDL model is needed for the clear formal description of the STP-ISS internal mechanisms and specification analysis [15]. The SDL specification is used as a separate document describing the specified mechanisms, and it is a useful part for the main protocol specification document.

The SystemC model shows the STP-ISS protocol operation over SpaceWire network, and it gives an ability to test the network configuration and test networking features [14].

The reference code is intended to be used as the reference for the programmers, who will implement STP-ISS in the on-board software. The reference code is a software implementation of the STP-ISS protocol in C++ language [16]. This implementation corresponds to the specification as accurate as it is possible. The C++ reference code describes the logical structure of the protocol, its interfaces and internal mechanisms. All methods, which describe protocol functionality, are provided with detailed comments for each line. In addition, in order to check and prove the accuracy of STP-ISS the model contains a number of test scenarios for studying and demonstration of protocol functioning. Each scenario launch produces detailed log files with event traces of nodes and of a channel.

This reference code is used for studying of the protocol functionality. Moreover, it could be translated into the other programming languages and used for the implementation of STP-ISS in the on-board software.

The other possible application of the reference code is an implementation of a tester, that could be useful for testing of the software models or hardware implementation of the protocol. In this case, reference code is used as a black box, which works with full conformance to the STP-ISS specification [17]. This reference implementation of the protocol could be placed on the one side of the connection, and the software model or hardware protocol implementation – on the other side. That software or hardware implementation is called Device Under Test (DUT). Reference code can generate different types of packets and the DUT should respond to them. Depending on the result of the data exchange we can make a decision, if the DUT works in conformance to the specification or not.

V. Software-to-Hardware Tester

We used the reference code to implement a Software-to-Hardware Tester (S2HT) for the STP-ISS protocol. This title means that we test the real on-board *hardware* with the *software* implementation of a protocol model (reference code).

And this is a software conformance tester, if we refer to the overview from the chapter II.

Software part of the S2HT consists of the following parts:

- Test engine "stp_testengine", containing a set of testing scenarios;
- STP-ISS reference code "stp_reference";
- Error generation module "error_generator".

Test engine is a set of testing scenarios for checking of correctness of the testing equipment operation. This module is implemented in SystemC, which represents a simulation library of C++ programming language [18]. After the start of the tester operation the user is able to choose the number of a test scenario and a test starts to execute. In the course of the test the S2HT performs a fixed number of actions according to the particular scenario, for example, protocol configuration or transmission of different types of packets. When the test is completed the tester displays the results. During execution of the test, the tester gathers the information on test operation and different events to the log files.

Error generation module is implemented for testing of the non-nominal cases in the communication process. Similarly the Test Engine module it is implemented in SystemC. This module gets data from the STP-ISS and can inject errors into a valid packet depending on a testing scenario. Error generation module is able to:

- Distort the transmitting data;
- Delete the service packets;
- Delete the EOP/EEP symbols;
- etc.

This module is also responsible for sending and receiving data from a hardware driver.

STP-ISS reference code part of the tester is a reference implementation of STP-ISS protocol with some modifications to the network level interface. Modifications were made so that the reference code can intercommunicate with the Error Generation module: we changed network interface primitive functions from serial byte to full packet transfer. It was necessary for data preparation for passing to the hardware driver. These modifications give an ability to work with Star-Dundee USB Brick and SpaceWire-Ethernet drivers. In the tester implementation the reference code is a separate library that is used by the software.

The general architecture of the implemented Software-to-Hardware tester is shown in Fig. 5.

This software part of the tester is installed on the PC. Current version of S2HT operates under the Ubuntu operation system.

PC should be connected to the DUT via the SpaceWire cable. It is possible to use any suitable SpaceWire hardware (Hardware in Fig. 5) for connection of the PC to the SpaceWire device. For example it can be Star-Dundee SpaceWire Brick Mk2 [19] or Ethernet-SpaceWire Bridge [20]. The Ethernet-SpaceWire Bridge could be used to connect SpW network through Ethernet interface to end user which is especially useful in testing purposes.

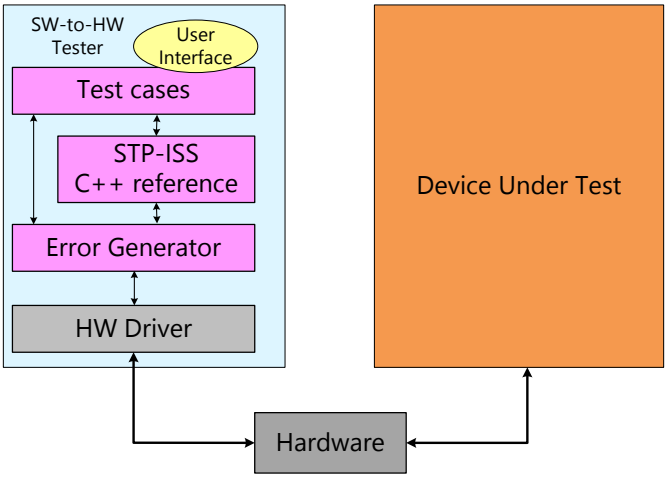


Fig. 5. Software-to-Hardware tester architecture

Each of these devices (brick or bridge) provides a special driver (HW Driver in Fig. 5) with an API for sending and receiving SpaceWire packets, time-codes and interrupt-codes.

The other side of the connection is the DUT. This device should have the SpaceWire port and should satisfy the following general requirements:

- implementation of STP-ISS at least rev. 1;
- SpaceWire packets sending and receiving functionality;
- indication of data packet or command reception;
- implementation of a SpaceWire link interface.

The DUT can be represented by the following devices:

- real on-board equipment which is a "black-box" in the sense that we do not have a model of it, thus, can rely only on its observable input/output behavior;
- PC with the SpaceWire interface (including a special SpaceWire networking board).

If we use another PC as the DUT, we have some additional abilities for testing and verification. We can set up the reference code of STP-ISS to DUT and observe, how both sides of the connection communicate with each other via the SpaceWire link. The other beneficial option is to test the real VHDL implementation of STP-ISS IP Core [20]. Fig. 6 shows a way of connection of STP-ISS IP block to the S2HT.

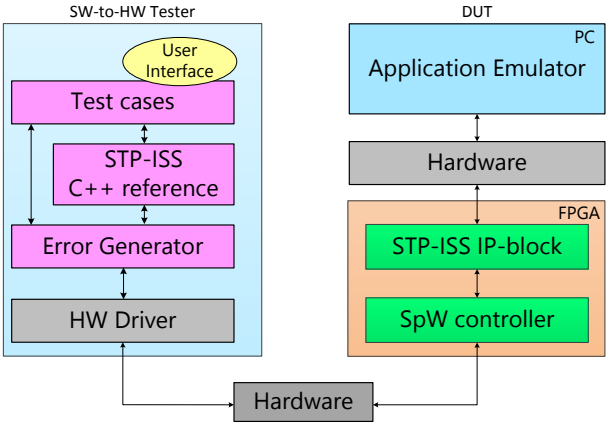


Fig. 6. STP-ISS IP Core interconnection with S2HT

Fig. 7 and Fig. 8 show the Software-to-Hardware tester that is implemented in our laboratory. This tester consists of a laptop with a pre-installed Ubuntu OS and Test Software. This laptop could be connected to the DUT by means of Ethernet-SpaceWire Bridge (see Fig. 7) or SpaceWire Brick Mk2 (see Fig. 6).

Fig. 7. SUAI Software-to-Hardware tester for STP-ISS connected with DUT via SpaceWire Brick Mk2

Fig. 8. SUAI Software-to-Hardware tester for STP-ISS connected with DUT via Ethernet-SpaceWire Bridge

VI. Application of Software-to-Hardware Tester

A. S2HT for STP-ISS Protocol Testing

Current version of the Software-to-Hardware tester is able to test the following mechanisms of STP-ISS:

- assembling and disassembling of STP-ISS user data packets and service packets;
- data transmission mechanisms;
- best-effort quality of service;
- guaranteed quality of service;
- error detection and recovery mechanisms;
- SpaceWire time-codes transmission and reception.

Testing should focus not only on normal protocol operation checking, but also on operation in exceptional and critical situations.

There is a number of STP-ISS mechanisms the Software-to-Hardware tester is not able to test:

- Receiving and Transmitting of SpaceWire distributed interrupts and interrupt acknowledges, which are not supported by a SpaceWire Brick Mk2;
- Settings of configuration parameters for DUT;
- Any problems in SpaceWire link-level functionality and other SpaceWire equipment errors (e.g. SpaceWire cable and Brick Mk2), because it is out of S2HT scope.

S2HT provides two alternatives for the user interface: console application and graphical user interface. Fig. 9 shows launched S2HT graphical user interface with a selected test scenario #2.

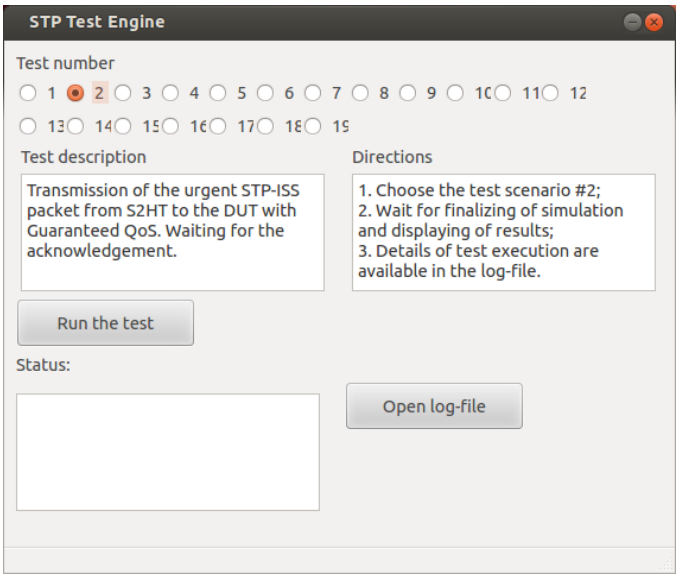


Fig. 9. Execution of the test scenario #2

Test scenario #2 is intended to check assembling and disassembling of STP-ISS user data packets and service packets mechanisms of the DUT. The tester sends a guaranteed urgent packet to the DUT and waits for an acknowledgement. If a correct acknowledgement is received, then this mechanism is correctly implemented inside the DUT.

During the S2HT exploitation the user should be always sure that all the equipment is correctly connected and configured. Moreover, if DUT is not able to send data, thus some of the test scenarios could not be executed successfully, because the tester needs a response form the remote side of the connection.

B. S2HT for SpaceWire Oriented Protocols Testing

Although S2HT is aimed for STP-ISS transport protocol testing, it can be applied for other transport protocols operating over SpaceWire. Figure 9 shows the variety of different applications of S2HT.

In order to get a Tester for another protocol it is necessary to implement the transport protocol in C++. An interface with the Error Generator module will stay the same as for STP-ISS protocol, but the upper interface with Test Cases module will be different depending on the protocol.

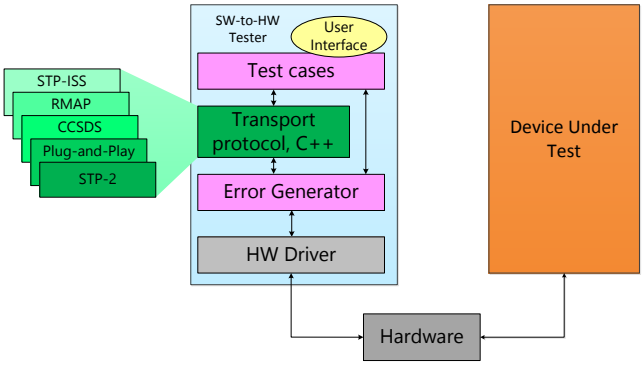


Fig. 10. Application of the S2HT for other protocols testing

The DUT, in turn, should be able to operate in accordance with the tested protocol. Test cases module should be updated in order to perform appropriate conformance testing.

CONCLUSION

The implemented Software-to-Hardware tester is a promising tool that could help the developers to ensure that STP-ISS equipment operates correctly. The implemented list of testing scenarios should give the full test coverage for the testing devices, so the result of the tester exploitation should be simple – true or false. That means, did the DUT successfully passed all the tests or not. If there are any faults in particular test scenarios, then the developer could analyse log-files and find out, which mechanism is implemented incorrectly.

Current implementation of a tester is able to test the equipment that operates in conformance with STP-ISS protocol specification rev.1. So the work that is still need to be done in this field is updating the tester to the 2nd STP-ISS revision conformance.

Finally, S2HT tester architecture and modules can be used for implementation of a Tester for SpaceWire oriented transport protocols.

ACKNOWLEDGEMENT

The research leading to these results has received funding from the Ministry of Education and Science of the Russian Federation under the contract RFMEFI57814X0022.

REFERENCES

[1] P. Marwedel, *"Embedded System Design"*, Springer, 2006. 241 p.

[2] Martha Gray, Alan Goldfine, Lynne Rosenthal, Lisa Carnahan. "Conformance Testing", *National Institute of Standards and Technology*, Gaithersburg, USA, 2010.

[3] Tektronix official website, HDMI compliance test software, Web: http://www.tek.com/datasheet/product-software/options-hdm-hdm-ds-hdm-dsm-ht3-and-ht3-ds-datasheet-1.

[4] Rohde&Schwartz official website, R&S®CMW - Conformance testing solution for eCall/ERA-Glonass, Web: https://www.rohde-schwarz.com/en/applications/r-s-cmw-conformance-testing-solution-for-ecall-era-glonass-application-card_56279-106883.html.

[5] A. Krupp, W. Muller "A Systematic Approach to the Test of Combined HW/SW Systems", in *Proc. Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Paderborn University / C-LAB, Paderborn, Germany, 2010, pp. 323 – 326.

[6] H. Kahlouche, C. Viho, M. Zendri "Hardware Testing Using a Communication Protocol Conformance Testing Tool", *TACAS/ETAPS'99, LNCS 1579, Springer-Verlag*, Berlin Heidelberg, 1999, pp. 315-329.

[7] Star-Dundee website, SpaceWire Conformance Tester, Web: https://www.star-dundee.com/products/spacewire-conformance-tester.

[8] Koopman, P., Chakravarty, T. "Cyclic *redundancy code (CRC) polynomial selection for embedded networks". DSN '04 Proceedings of the 2004 International Conference on Dependable Systems and Networks*, IEEE Computer Society, 2004. pp. 145-154.

[9] Tanenbaum, A. S., *Computer Networks*, Fifth Edition; Prentice Hall, 2011. 962.

[10] Tomlinson, R.S., "Selecting Sequence Numbers", *Proceedings of the ACM SIGCOMM/SIGOPS Interprocess Communication Workshop, and ACM Operating Systems Review*, Vol. 9, No. 3, July 1975, Association for Computing Machinery, New York, 1975.

[11] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, D. Dymov "STP-ISS Transport Protocol for Spacecraft On-board Networks", *Proceedings of 6th International Conference SpaceWire 2014 Program*, Athens, Greece, 2014, pp. 26-31.

[12] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, S. Kochura, S. Openko, D. Dymov "Second Revision of the STP-ISS Transport Protocol for On-Board SpaceWire Networks", *Proceedings of 17th Conference of Open Innovations Association FRUCT*. Yaroslavl: Russia, 2015. pp.192-200.

[13] D. Dietterle, *"Efficient Protocol Design Flow for Embedded Systems"*, PhD thesis in Computer Science. Cottbus, 2009.

[14] Y. Sheynin, V. Olenev, I. Lavrovskaya, I. Korobkov, S. Kochura, S. Openko, D. Dymov "STP-ISS Transport Protocol Overview and Modeling", *Proceedings of 16th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications (FRUCT) Program*. Oulu: University of Oulu, 2014. pp.185-191.

[15] P. Morozkin, I. Lavrovskaya, V. Olenev, K. Nedovodeev, "Integration of SDL Models into a SystemC Project for Network Simulation", in F. Khendek et al. (Eds.), *SDL 2013: Model-Driven Dependability Engineering, Lecture Notes in Computer Science, Volume 7916* (pp. 275-290). Berlin: Springer Berlin Heidelberg, 2013.

[16] B. Stroustrup, *The C++ Programming Language*, 4th Edition. USA: Addison-Wesley, 2013.

[17] M. Krichen, S. Tripakis, "Black-Box Conformance Testing for Real-Time Systems", *11th international SPIN workshop on model checking of software (SPIN'04)*. LNCS, vol. 2989. Springer, Berlin.

[18] D. Black, J. Donovan, B. Bunton, A. Keist, *"SystemC: From the Ground Up"*, NY: Springer, 2010.

[19] Star-Dundee website, SpaceWire-USB Brick Mk2, Web: https://www.star-dundee.com/products/spacewire-usb-brick-mk2.

[20] Yablokov, E., Rozanov, V., Vinogradov, A. "Protocol for Connection Ethernet Interface to SpaceWire Networks", *Proceedings of 17th Conference of Open Innovations Association Finnish-Russian University Cooperation in Telecommunications FRUCT*. Yaroslavl: Russia, 2015, pp. 362-367.

[21] A. Ben Abdallah, *"Multicore Systems On-Chip: Practical Software/Hardware Design"*, Second Edition. Atlantic Press, 2013.

Papers Indexed by Author

Author Surname A – K

Author Surname L - O

Author Surname P - Z

Papers Indexed by Session

Tuesday 25th October

Missions & Applications (Short Papers)

SpaceFibre 1 (Long Papers)

Networks & Protocols (Short Papers)

Components (Short Papers)

Wednesday 26th October

Missions & Applications (Long Papers)

Test & Verification (Short Papers)

SpaceFibre (Short Papers)

Networks & Protocols 1 (Long Paper)

Poster Presentations

Thursday 27th October

Components (Long Papers)

Networks & Protocols 2 (Long Papers)

SpaceFibre 2 (Long Papers)

Test & Verification (Long Papers)

Exhibitors

AXON' CABLE

The Axon' group designs and manufactures wire, cable, connectors and cable assemblies for advanced technology applications in the principal fields of space, aeronautics, medical electronics, automotive and scientific research. Headquartered in France (100 Km east of Paris) the Group employs some 1700 staff in 14 subsidiaries across Europe, America and Asia, with an annual turnover of €115 million euro.

Axon' Cable has been involved in many space projects, including the International Space Station, various LEO and GEO satellites and rocket launchers including Ariane 5, and can boast flight heritage dating back to 1997.

The group offers various types of products for space applications:
- ESCC approved wires, cables and connectors,
- lightweight aluminium round cables and braids,
- aluminium bus bars for satellite power distribution,
- MIL-STD-1553 databus looms for digital transmission systems,
- high data rate links for Voice-Data-Image transmission including SpaceWire, IEEE1394, Ethernet and Fibre Channel,
- solutions suitable for the forthcoming multi-gigabit protocol, SpaceFibre,
- and custom-designed products for specific applications.

Additionally, Axon' has been involved either as prime or subcontractor on a number of ESA EMITS tenders including the development of high temperature thruster cables, the development of low mass SpaceWire, the evaluation of shielding techniques for Spacecraft harnesses, the evaluation of Nano-D for Space, the development of Combo Micro-D's and the provision of cables for the SpaceFibre Demonstrator.

4LINKS Limited

4Links test and simulation equipment for SpaceWire will save you time, delay, risk, and money. It does exactly what test equipment needs to do. It has proved to be interoperable with every design that it has connected to, while detecting faults including many not found by other methods.

Our solutions provide information to resolve faults, including longstanding ones, and often without the need to reproduce the fault. And the same hardware can be used - for devices, subsystems and complete satellites - at all stages of a mission development. The same innovative design, quality and support extends to 4Links' SpaceWire chips and SpaceWire IP. 4Links SpaceWire test and simulation equipment is reliable, accurate and excellent value. This is why more and more users are specifying 4Links as their SpaceWire products of choice.

ÅAC MICROTEC

ÅAC Microtec develops and supplies highly capable components, sub-systems and small satellite platforms. End-users include operators of commercial, R&D and educational space missions to whom reliability and resilience of the spacecraft are important. Thanks to our design approach and system architecture, platforms can swiftly be customized to meet specific mission and payload needs. Delivering customer data with high assurance and reliability is ÅAC Microtec's hallmark. Our fault-tolerant systems combine affordable performance with high mission confidence. For high-end payloads, our satellite solutions are the first choice of operators worldwide.

ÅAC has a strong competitive advantage in that it operates in the high-end segment of the small satellite market, and that the offered products are ITAR free. The products are flight proven and has strong heritage. Examples of products are On-board computers (OBCs), Mass Memory Units (MMUs/TCM), Power Control & Distributing Units (PCDUs), and Bluestone. Bluestone is an efficient way of distributing image data from satellites.

Contact info

ÅAC Microtec AB
Uppsala Science Park
Dag Hammarskjölds väg 48
SE-751 83 Uppsala
Sweden
Phone: +46 18-560130
info@aacmicrotec.com

COBHAM SEMICONDUCTOR SOLUTIONS

Cobham Semiconductor Solutions provides HiRel standard products, ASICs, and radiation testing services. Our Cobham Gaisler site in Goteborg, Sweden provides IP cores and supporting development tools for embedded processors based on the SPARC architecture along with SpaceWire Routers and boards.

The key product is the LEON synthesizable processor model together with a full development environment and a library of IP cores (GRLIB). Our personnel have extended design experience, and have been involved in establishing European standards for ASIC and FPGA development. Cobham Gaisler has extensive experience in the management of ASIC development projects, and in the design of flight quality microelectronic devices. The company specializes in digital hardware design (ASIC/FPGA) for both commercial and aerospace applications.

SHIMAFUJI ELECTRIC

Since 1990, Shimafuji Electric has been developing microcomputer boards including transmission, graphics and other complex peripheral functions and also producing small number of products for some OEMs.

Shimafuji have joined the Japan SpaceWire user Group since early days. We developed the SpaceWire compliant cubic computer - Space Cube with JAXA, and we have some SpaceWire function boards, like the Universal FPGA Board, The Sampling ADC, The Digital I/O, and ETC since 2005. Then, our one of latest model is the 4 port Space Wire to Gigabit Ether R2 Unit and we are developed the 24-link SpaceWire Packet Recorder and 48-port SpaceWire Packet Generator based on the 12-slots microTCA SpaceWire Backplane system. We also developing the Grand Use SpaceWire for Industries.

In this year, we opened The SpaceWire Test Lab facility in our office for small space businesses, students and anyone who are interesting SpaceWire. This lab has clean booth, high function instruments, and off coarse SpaceWire testing instruments etc.

STAR-DUNDEE LTD.

STAR-Dundee is an aerospace engineering company, which designs network and related data-handling technology for use on-board spacecraft. STAR-Dundee provides electronic test and development equipment and chip designs for spaceflight applications.

Our highly experienced engineers were instrumental in the development of SpaceWire, writing the ECSS standard with inputs from international spacecraft engineers. SpaceWire is now widely used on-board spacecraft with over 100 space missions already in orbit or currently being designed using SpaceWire technology. Our engineers are currently leading the research, technical development and standardisation of the next generation of SpaceWire technology, SpaceFibre, which is a substantial leap forward, offering much higher data rates, quality of service, fault detection, isolation and recovery, deterministic data delivery, low latency time-synchronisation and event signalling, and many other features and benefits.

Since 2002, STAR-Dundee has provided SpaceWire evaluation, test and development equipment to the world's space agencies and aerospace companies. Our SpaceWire interface boards and units are used in Electronic Ground Support Equipment (EGSE) for integrating and testing many spacecraft. Our IP cores are integrated in spaceflight systems monitoring the Earth, exploring our Solar System, studying the universe and supporting commercial space applications.

STAR-Dundee is committed to providing the best possible solution for your application. Our team of highly qualified and experienced engineers understands the challenges of designing systems for space applications. Our well proven technology has flown on many high profile space missions. Part of our commitment to our customers is the effort that we spend on the research, development and standardisation of data-handling technology. SpaceFibre is the latest manifestation of our commitment to engineering excellence and international standardisation.

GLENAIR

Glenair – out of this world of interconnect solutions

SpaceWire cable assemblies:

Glenair offer a complete range of SpaceWire cable assemblies for laboratory and flight use.
In support of the SpaceWire protocol Glenair also offer a complete range of Micro D connectors for vacuum chamber and router interface use.
For more information on Glenair's space products portfolio please contact:

Ross Thomson, Business Manager - Space Interconnect Systems
Glenair UK Ltd
40 Lower Oakham Way
Oakham Business Park
Mansfield, Nottinghamshire
NG18 5BY, UK
e-mail: rthomson@glenair.co.uk
Office: +44 (0) 1623 638100
Mobile/ cell: +44 (0) 7711 029 715

SpaceFibre:

Glenair designs and manufactures a full range of fiber-optic interconnect products to support spacecraft systems.
These include radiation-tolerant high-speed opto-electronic transceivers supporting SpaceFibre, sRIO and other high-speed protocols up to 10 Gbps per lane, as well as fiber-optic cable assemblies, connectors, inspection and cleaning kits, and training of personnel to insure mission success.
For more information on Glenair's fibre optic product portfolio and capability please contact:

Ronald T. Logan Jr., Ph.D.
Chief Technologist, Sr. Director Active Components
Glenair Inc.
1211 Airway,Glendale
California 91202 -2497, USA
e-mail: rlogan@glenair.com
Office: +1 818 247 6000

www.glenair.com

INGENIARS S.R.L

IngeniArs S.r.l. is a spin-off company of the University of Pisa born in May 2014, built upon the large experience (more than 20 years) of its co-founders. Main focus of the company is the space business with a wide range of services and products such as:

- Specific products for design and validation of on-board high speed data communications based on SpaceWire and SpaceFibre standards
- Development of ad-hoc on-board data processing and data-handling HW/SW systems
- Development of ad-hoc Electrical Ground Support Equipments (EGSEs) and Validation Platforms for Space Equipment

As far as SpaceWire/SpaceFibre system is concerned IngeniArs offers an extensive portfolio of products (http://www.ingeniars.com/english/products/space.html) allowing the final user to design and validate systems based on such standards. In particular IngeniArs products portfolio is composed of:

- SpaceWire Codec, Spacewire Router and SpaceFibre Codec IP Cores
- Spacewire/SpaceFibre Link Analyser (Stand-Alone Real Time Validation Platform)
- Spacewire/SpaceFibre NI PXI Link Analyser (Real Time Validation platform fully compatible with National Instrument Platforms based on PXI interfaces)

Despite its recent foundation IngeniArs already counts important customers such as Finmeccanica-Leonardo and Thales Alenia Space as well as a strategic partnership with National Instrument for development of EGSEs and validation systems. IngeniArs was also recently awarded with contracts as prime by ESA and H2020 SME instrument.

Further information about IngeniArs S.r.l. can be found at http://www.ingeniars.com/.

MITSUBISHI ELECTRIC CORPORATION

Mitsubishi Electric's space technology includes the manufacture and implementation of satellites, satellite components, and ground systems. Over the past four decades, we have completed more satellite projects for communications concerns, government agencies, and other large-scale clients than any other Japanese company, making Mitsubishi Electric the leading company for space systems in Japan. We have a distinct advantage when it comes to designing, building, launching and controlling satellites, because we also excel in the solar panel, antenna, amplification, tracking, control and ground station system technologies that make satellites practical to own and operate.

http://www.mitsubishielectric.com/

MITSUBISHI HEAVY INDUSTRIES, LTD.

Our Technologies, Your Tomorrow

MITSUBISHI HEAVY INDUSTRIES, LTD

More than 130 years have passed since Mitsubishi first leased the Nagasaki Shipyard from the government's Ministry of Industry in 1884. The technologies of the MHI Group supported Japan through unbridled changes in its quest for modernization and globalization.
The Group uses the technological foundation accumulated over these long years to provide products and innovations in a wide range of fields. In 2014, the company completed its transition to a domain system, achieving even greater synergy and contributing even more to the development of society.

As Japan's leading defense and space systems integrator, the Integrated Defense & Space Systems Domain combines the technology and expertise of each of its businesses, resulting in a system that makes it possible to coordinate land, sea and air defense initiatives, as well as reinforcing MHI's international competitiveness in the space industry.

MHI provides launch services with the H-IIA, Japan's primary launch vehicle, and has also participated in the development and production of KIBO, the Japanese Experiment Module (JEM) on the International Space Station, contributing to space development in Japan.

https://www.mhi-global.com/company/aboutmhi/outline/index.html

\Orchestrating a brighter world

NEC

NEC CORPORATION

NEC is a multinational provider of information technology and network solutions & products to business enterprises, communications services providers and government agencies since established in 1899. In addition, NEC is a few companies which have both Space technology and ICT. NEC has more than 50 years of expertise in space business, and has been providing wide diversity of space products including various satellite systems and optical and radio wave sensors.

We also offer solutions of ICT which use data from sensors. NEC contributes to advanced urban development with biometric identification technology, versatile sensing technology and analytic technology that makes high-precision forecasting and prediction possible. NEC also leverages sensing technologies and big data analysis technologies to support the advancement of lifelines through ICT.

As the industry makes the shift from Space development to Space utilization, NEC's space business intends to transform itself into an enterprise that provides space solutions. NEC can offer space solutions by fusing space technologies and IT/Network technologies such as for remote sensing area which is the focus of increasing attention. The NEC space solution aims to provide information services that can provide "any" user with "any" information that need "anytime" and "anywhere" by processing, formatting and storing both the observation/survey data acquired from space systems and the various kinds of sensor data collected from terrestrial sources.

Via the space solution, NEC contributes to realize an information society friendly to human and the earth.

For more information, visit NEC space system solutions at:
http://www.nec.com/space/

JAXA esa NASA ROSCOSMOS Orchestrating a brighter world NEC MITSUBISHI ELECTRIC Changes for the Better MITSUBISHI HEAVY INDUSTRIES, LTD. OSAKA UNIVERSITY UNIVERSITY OF DUNDEE