Issue 68
Third Quarter 2009

# Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

## Accelerate Innovation with Targeted Design Platforms

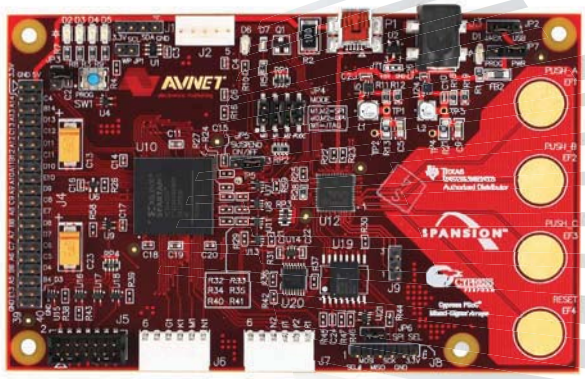
INSIDE



XILINX®
www.xilinx.com/xcell/

# Xilinx® Spartan®-3A
## Evaluation Kit

DESIGNED BY AVNET



The Xilinx® Spartan®-3A Evaluation Kit provides an easy-to-use, low-cost platform for experimenting and prototyping applications based on the Xilinx Spartan-3A FPGA family. Designed as an entry-level kit, first-time FPGA designers will find the board's functionality to be straightforward and practical, while advanced users will appreciate the board's unique features.

BEHIND THE WHEEL™

Get Behind the Wheel of the **Xilinx Spartan-3A Evaluation Kit** and take a quick video tour to see the kit in action *(Run time: 7 minutes)*.

## Ordering Information

| Part Number | Hardware | Resale |
|---|---|---|
| AES-SP3A-EVAL400-G | Xilinx Spartan-3A Evaluation Kit | $39.00* USD (*Limit 5 per customer) |

**Take the quick video tour or purchase this kit at: www.em.avnet.com/spartan3a-evl**

## Target Applications

- General FPGA prototyping
- MicroBlaze™ systems
- Configuration development
- USB-powered controller
- Cypress® PSoC® evaluation

## Key Features

- Xilinx XC3S400A-4FTG256C Spartan-3A FPGA
- Four LEDs
- Four CapSense switches
- I²C temperature sensor
- Two 6-pin expansion headers
- 20 x 2, 0.1-inch user I/O header
- 32 Mb Spansion® MirrorBit® NOR GL Parallel Flash
- 128 Mb Spansion MirrorBit SPI FL Serial Flash
- USB-UART bridge
- I²C port
- SPI and BPI configuration
- Xilinx JTAG interface
- FPGA configuration via PSoC®

## Kit Includes

- Xilinx Spartan-3A evaluation board
- ISE® *WebPACK*™ 10.1 DVD
- USB cable
- Windows® programming application
- Cypress MiniProg Programming Unit
- Downloadable documentation and reference designs

XILINX®

AVNET® electronics marketing

Avnet Green Initiative

NEDA Supplier Authorized Distributor

*Accelerating Your Success™*

**1.800.332.8638**
**www.em.avnet.com**

Xcell journal

PUBLISHER Mike Santarini
mike.santarini@xilinx.com
408-626-5981

EDITOR Jacqueline Damian

ART DIRECTOR Scott Blair

DESIGN/PRODUCTION Teie, Gelwicks & Associates
1-800-493-5551

ADVERTISING SALES Dan Teie
1-800-493-5551
xcelladsales@aol.com

INTERNATIONAL Melissa Zhang, Asia Pacific
melissa.zhang@xilinx.com

Christelle Moraga, Europe/
Middle East/Africa
christelle.moraga@xilinx.com

Yumi Homura, Japan
yumi.homura@xilinx.com

SUBSCRIPTIONS All Inquiries
www.xcellpublications.com

REPRINT ORDERS 1-800-493-5551

Xcell PUBLICATIONS
www.xilinx.com/xcell/

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3400
Phone: 408-559-7778
FAX: 408-879-4780
*www.xilinx.com/xcell/*



# Golden Age of Semiconductors Points to Platinum Era Ahead

## 'Only the beginning' of a great new wave of invention built on the IC.

Many years ago, one of my former colleagues over at *EDN* magazine mailed me a copy of Gordon Moore's landmark article "Cramming More Components onto Integrated Circuits." It was a reprint of a piece that first appeared in the now-defunct magazine *Electronics* on April 19, 1965, and I've kept it at my desk ever since. If you've never read the article, do so. Moore opens with the sentence, "The future of integrated electronics is the future of electronics itself," and then goes on to make a slew of brilliant predictions, foreseeing the inventions of the personal computer, the Internet and the handheld market—fruits of the ever-progressing IC.

Of course, the article is considered a landmark chiefly because Moore goes on to write that the number of transistors on an IC will double roughly every year (later amended to every two years), a formulation that Cal Tech professor Carver Mead famously dubbed "Moore's Law." The semiconductor and, in turn, electronics industries have soared for the last 44 years paced indisputably by Moore's Law.

Back in 2005, I had the pleasure of covering a Computer History Museum event for *EDN* where Carver Mead actually interviewed Moore about his law. (You can view the video of their chat at *http://www.youtube.com/watch?v=MH6jUSjpr-Q&feature=PlayList&p=6B12A0FACFA35D1F&index=7*.) More recently, I witnessed another event of historic importance for the industry: the induction ceremony of 15 semiconductor giants into the National Inventors Hall of Fame (read my *pldesignline.com* blog coverage at *http://www.dspdesignline.com/news/217400639*).

Moore and Mead were among the 15 inductees, but I was really there to honor Xilinx co-founder Ross Freeman, who was also inducted (posthumously) into the Hall of Fame, and to learn more about him from his colleagues and family. For those of you who may not be familiar with the name, Freeman invented the FPGA in 1984 (Patent No. 4,870,302), only five years before he died prematurely at the age of 45. At this fascinating event, I learned that the FPGA probably would not have been commercialized or grown as successful as it has had Freeman, his co-founders and, ultimately, Xilinx's early investors not wholeheartedly embraced Moore's Law.

*Xilinx co-founder Ross Freeman*

Essentially, Freeman's early FPGA circuit wasn't considered the most efficient use of transistors, but it was inventive and unique. In those days, transistors were very expensive to produce. Freeman's invention traded off transistor minimization and utilization for flexibility, fast turnaround and the convenience of outsourced manufacturing. Back in

1984, Freeman predicted that as time and Moore's Law progressed, FPGAs would keep pace, at least doubling in logic-cell counts, and that the cost per transistor would decline dramatically, making programmable devices more attractive to a growing number of users.

Just as Moore identified a trend in the cycle of IC manufacturing, Freeman—along with fellow co-founders Bernie Vonderschmitt and Jim Barnett—pinpointed a related economic trend. Namely, if manufacturing kept pace with Moore's Law by introducing a new process technology every two years, the associated costs of building a fab and outfitting it with the latest process equipment would, over time, make owning a fab less feasible for a growing number of chip companies.

So, for Xilinx's first production FPGA, the XC2064, Vonderschmitt masterminded a revolutionary deal to have Seiko manufacture the parts. That marked the birth of the fabless model. Today, the vast majority of semiconductor and system companies no longer manufacture their own ICs, but contract with foundries because, as Ross and Bernie predicted, manufacturing costs have spiraled with each new process introduction.


*Ross Freeman's mother, Ethel, holds the Xilinx co-founder's award at National Inventors Hall of Fame ceremony. Ross' brother Fred Freeman (right) accepted the award on the family's behalf.*

The exorbitant cost, which foundries pass on to customers, has now made even fabless production of ASICs and ASSPs economically out of reach for an ever-growing number of designers. These engineers are increasingly turning to FPGAs.

Today, Freeman's predictions about the increasing value of FPGAs resonate more strongly than ever, thanks in part to Xilinx's employees and customers, who have collaborated over these last two decades to advance FPGA technology in extraordinary ways, helping Xilinx innovate using the world's most advanced process technologies. As a result of this ongoing innovation, today's FPGAs contain hundreds of thousand of logic cells and include microprocessor cores, DSP slices and high-speed I/O. FPGAs are gaining broader use every day as designers increasingly choose them as the means to quickly turn their inventions into reality.

The spirit of innovation honored at the 2009 Inventors Hall of Fame ceremony wasn't just a celebration of the golden age of semiconductors. It was also a celebration of myriad inventions you designers will continue to enable in the future. The last 50 years may have been golden, but the next 50 could surely be platinum. As Cal Tech's Mead graciously put it in his brief acceptance speech, "this is only the beginning…"

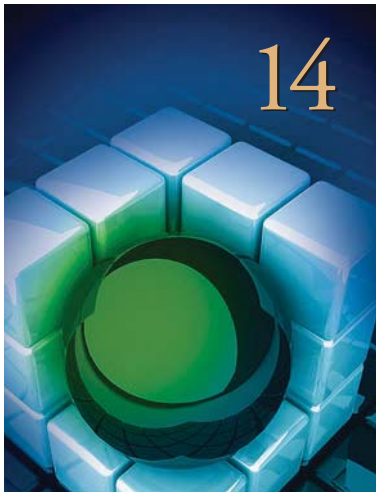
Mike Santarini
Publisher

## VIEWPOINTS

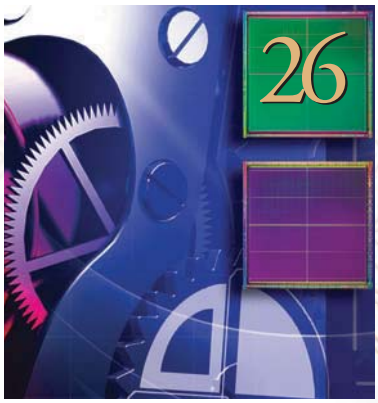


## XCELLENCE BY DESIGN APPLICATION FEATURES





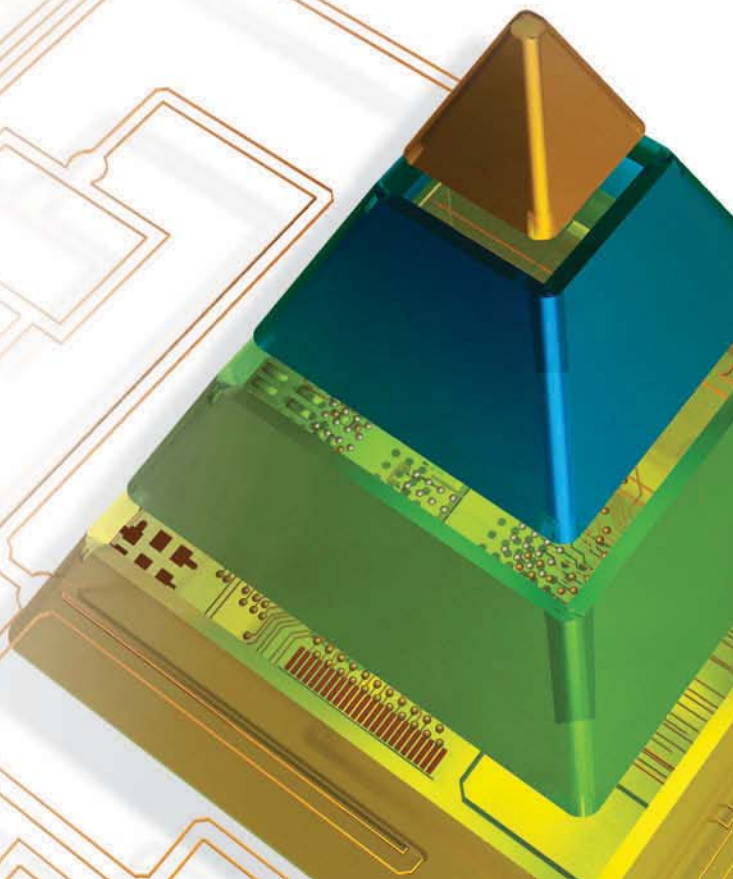

22





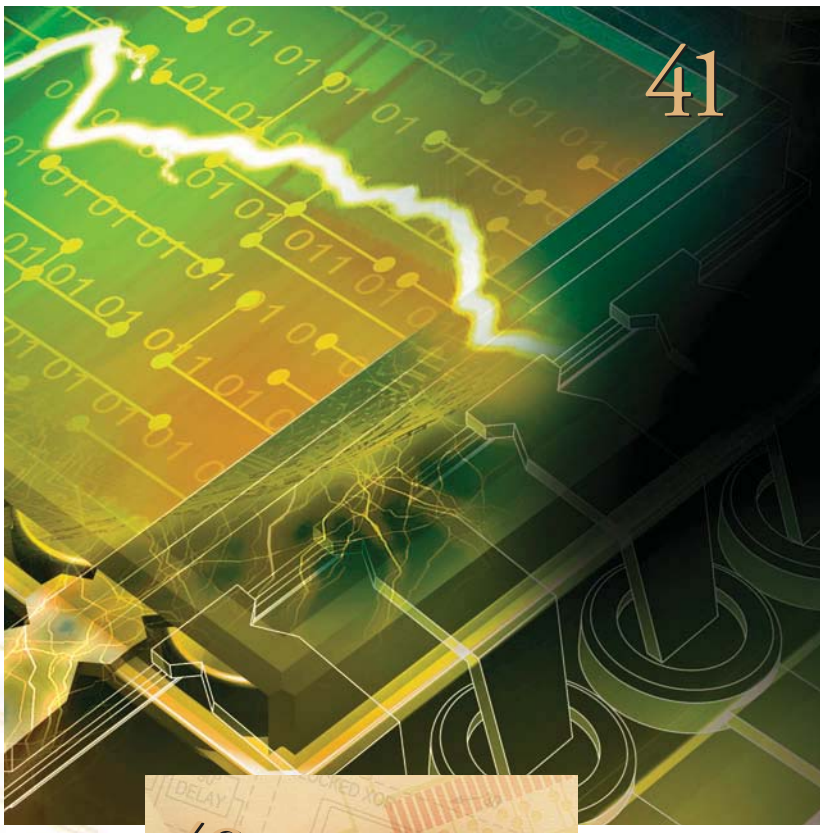

## Cover Story





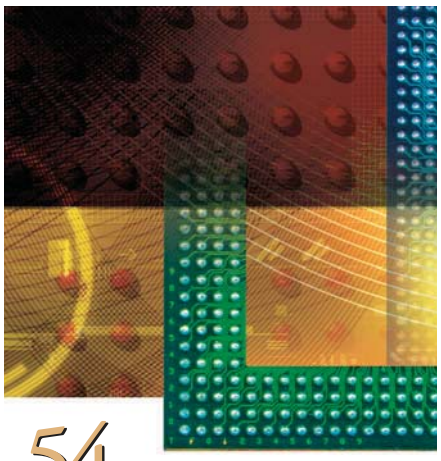
## THE XILINX XPERIENCE FEATURES






## XTRA READING



# Targeted Design Platforms Put Innovation on Fast Track

## Xilinx pioneers cutting-edge infrastructure for its latest FPGAs, the Spartan-6 and Virtex-6.

by Mike Santarini
Publisher, Xcell Journal
Xilinx, Inc.
*mike.santarini@xilinx.com*

When Moshe Gavrielov joined Xilinx in 2008 as the company's new president and CEO, he brought with him decades of knowledge as a semiconductor executive and designer of ICs, and the aggregate perspective that many economic realities are converging to make programmability an imperative for a greater number of electronic end products.

Prior to joining Xilinx, Gavrielov managed LSI Logic's ASIC group for many years before moving on to serve as the CEO for EDA firm Verisity. During his years in the business, he has witnessed firsthand the rise in both IC design complexity and cost of manufacturing, in tandem with ever-tighter development budgets as companies strive to rapidly address new markets and evolving standards.

'Targeted Design Platforms will facilitate the FPGA development process for our growing user base,' says Xilinx CEO Moshe Gavrielov, 'so designers can get the most out of our FPGAs and can get their innovations to market faster.'

Gavrielov concluded that these converging factors would present Xilinx with a major opportunity for growth. That's because FPGAs don't burden the users with manufacturing costs. Also, the devices are reprogrammable and, thus, inherently flexible and forgiving of design errors.

However, to really seize the opportunity, Xilinx not only has to provide customers with the best FPGAs in the industry, but must complement that silicon with world-class tools, intellectual property (IP) that's easy to integrate, evaluation kits, services, helpful documentation and targeted reference designs. Each part of the puzzle is necessary to facilitate customers getting their innovations to market quickly.

To turn Gavrielov's vision into a reality, the hard-working employees of Xilinx and its partners put their heads together to engineer the Xilinx® Targeted Design Platform approach, which streamlines the FPGA design process for all users, from novice to expert.

### The Xilinx Targeted Design Platform

To communicate this approach, Xilinx represents the strategy as a pyramid consisting of four layers: the Base Targeted Design Platform, the Domain-Specific Platform and the Market-Specific Platform, topped off with the capstone of user differentiation (Figure 1). Users can automate a greater percentage of the more-basic parts of their designs by adding, at their discretion, upcoming Xilinx domain-specific and market-specific platform offerings to the Base Targeted Design Platform. The approach will enable them to get products to market fast, and focus the majority of their design cycles on the elements that will really differentiate those products.

"Today, our high-performance Virtex®-6 and high-volume Spartan®-6 FPGAs boast several hundreds of thousands of programmable logic cells, up to 11.2-Gbit/second transceivers, 38 Mbits of block RAM and 2,000 DSP slices that designers can leverage to create innovations for an ever-growing number of applications," said Gavrielov. "All told, the opportunity is ripe for Xilinx to more rapidly take market share away from ASICs and ASSPs and assume a more central role at the heart of next-generation electronic innovations. Targeted Design Platforms will facilitate the FPGA development process for our growing user base, so designers can get the most out of our FPGAs and can get their innovations to market faster."

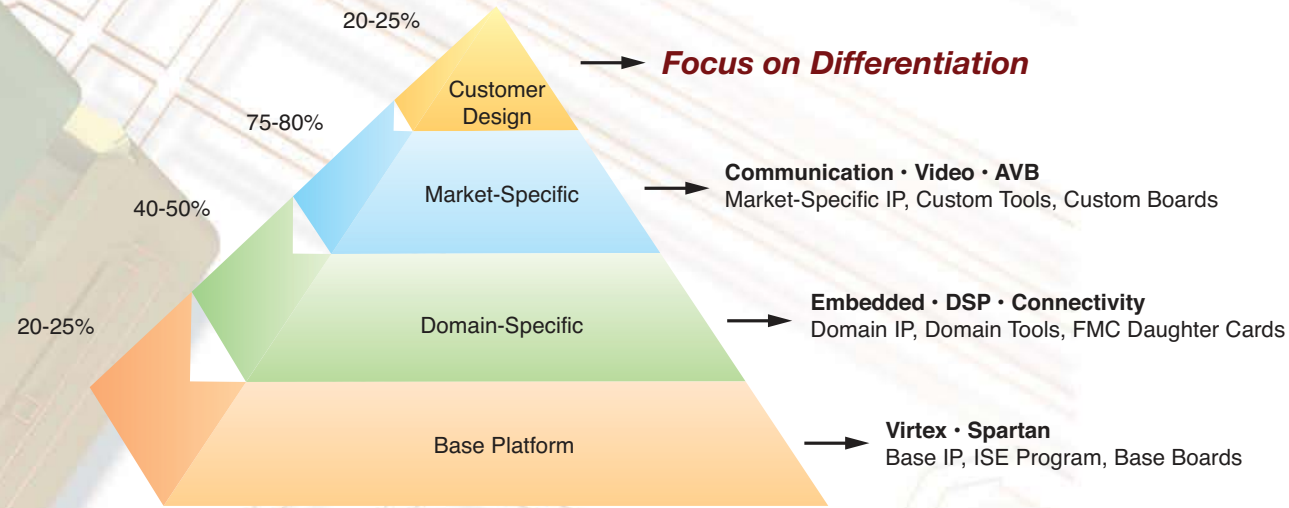



*Figure 1 – The Xilinx Targeted Design Platform squarely aims at increased user productivity.*

At the foundation layer of the pyramid is the Base Targeted Design Platform, which Xilinx made available in June of this year, following the February 2009 launch of the Virtex-6 (*http://www.xilinx.com/products/virtex6/index.htm*) and Spartan-6 (*http://www.xilinx.com/products/spartan6/index.htm*) FPGA families. In April, the company released its ISE® Design Suite Edition 11 in support of the Targeted Design Platforms (*http://www.xilinx.com/support/documentation/white_papers/wp307.pdf*).

| # | Feature |
|---|---|
| 1 | DDR2 |
| 2 | SPI x4, x1, Ext. x4 Configuration |
| 3 | SPI Header |
| 4 | Parallel Flash |
| 5 | 10/100/1000 Ethernet |
| 6 | USB UART |
| 7 | IIC |
| 8 | Clock, Socket, SMA |
| 9 | FMC LPC connector |
| 10 | LED |
| 11 | DIP Switch |
| 12 | Pushbutton |
| 13 | USB JTAG |
| 14 | 12-pin (8 I/O) Header |
| 15 | VCCint Voltage Selection Header |

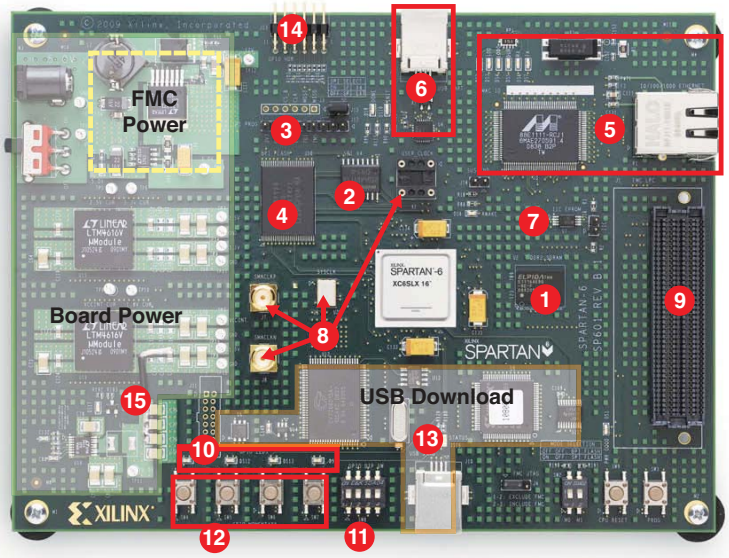



*Figure 2 – The Spartan-6 SP601 Evaluation Kit's board features a rich set of system functions.*

The June launch announced first delivery of the of the Base Targeted Design Platform for Virtex-6 and Spartan-6 FPGAs in the form of evaluation kits that combine all the elements customers need to realize the benefits this new platform approach promises. These evaluation kits combine the latest ISE Design Suite 11.2 with Virtex-6 LX240T and Spartan-6 LX16 FPGA evaluation boards, preverified base IP, base platform reference designs and a complete set of documentation that guides customers through the entire process, from setting up the hardware to installing software and beginning their design. (See *www.xilinx.com/products/targeted_design_platforms.htm.*)

"The release of the Base Targeted Design Platform means that Virtex-6 and Spartan-6 are now open for business," said Brent Przybus, director of product marketing. "Customers can now download ISE Design Suite 11.2, order base evaluation kits for Spartan-6 and Virtex-6 FPGAs, and access comprehensive documentation to begin developing designs."

### The Targeted Design Platform Methodology

With the Targeted Design Platform methodology, Xilinx users will start their projects with the Base Targeted Design Platform, gaining access to basic functions common in almost all applications. Then, depending on the design domain in which they typically work—logic designer, DSP, embedded-software programmer or systems engineer—they can add Xilinx Domain-Specific kits of their choice to the Base Targeted Design Platform.

These kits will provide domain-specific tools (available in ISE Design Suite 11 Editions), along with libraries of preverified domain-specific IP from Xilinx and key IP partners as well as domain-specific daughter cards. Designers can plug the cards into their Base Targeted Design platform evaluation boards by means of an FPGA Mezzanine Connector (FMC) on the base platform evaluation boards to more quickly implement their designs (see sidebar, page 12). Each Domain-Specific kit will also include targeted reference designs to help users quickly implement a greater number of functions in their designs.

In addition to these Domain-Specific kits, Xilinx and its partners are also developing Market-Specific Platforms, which will allow users to attach daughter cards geared for specific market segments to the base platform via the FMC connection, or to access market-specific boards. Each of the individual market-specific kits will likewise include preverified IP and reference designs to help users quickly add functions to their designs.

Przybus said that the more elements of the Targeted Design Platform users add, the faster they will complete their designs. By simply leveraging all that's available in the base platform, he said, users could conceivably implement up to 25 percent of their overall project, saving valuable time in which to develop the rest of their design. And if they leverage the domain-specific kits too, they should be able to quickly implement as much as 50 percent of their design. Going a step further and using the market-specific kits, designers could quickly implement up to 75 percent of their designs, Przybus said, freeing them to focus on the 25 percent of the design that will bring the greatest differentiation to the end products.

Przybus notes that some users may want to design their entire FPGA implementation from scratch, and they still can do so if they wish. But he believes a vast majority will welcome the great benefits of leveraging all the elements of the Targeted Design Platform.

### Targeting Your Needs

Przybus explains that Xilinx created Targeted Design Platforms to help every category of customer, from FPGA design experts who have used programmable devices religiously for many years; to ramping designers who may have traditionally focused on ASIC and ASSP designs but have started to transition the bulk of their efforts to FPGAs; all the way to novices who have never used FPGAs before but have traditionally targeted ASICs, or perhaps only have experience programming standalone processors.

With Targeted Design Platforms, Przybus said, Xilinx is giving the novices not only the FPGAs, IP and tools they will need to quickly start designing, but also reference designs to help them build their products.

"They can take these reference designs and add their own content to create differentiated products," said Przybus. Targeted Design Platforms will help novices quickly become familiar with FPGA design, and as they grow more experienced they can add even greater amounts of their own differentiation.

For those already knowledgeable about FPGA design and who are escalating their FPGA design efforts, Przybus said the Targeted Design Platforms will help accelerate their development time and get to market faster. "These groups are typically familiar with FPGA design, but one of the biggest issues they have is migrating a design, or components of a design, to a new FPGA," he said. "The Targeted Design Platforms will make that migration very simple, with reference designs that show how the new features are used, as well as useful migration techniques. They will be able to see how Xilinx did it, learn from that example and then accelerate their own design efforts."

Expert FPGA designers are also concerned with migrating their designs, but they are typically more focused on getting maximum performance out of their FPGAs or maximizing power efficiency, Przybus said. "The Targeted Design Platforms will provide these users with examples to help them get the greatest efficiencies out of their FPGA designs, leveraging the new FPGA and tool capabilities," he said. "We are providing them with a design environment that will allow them to closely monitor power consumption to optimize their designs for power efficiency, and also implement and analyze multigigabit transceivers and DSP slices running at their fastest rates. And for the first time, we are giving them a whole system-centric design to evaluate those elements."

The combination, Przybus said, "will accelerate their development time."

## Spartan-6 and Virtex-6 Evaluation Kits

A central element of the Base Targeted Design platform is the evaluation kit—specifically, the Spartan-6 SP601 Evaluation Kit and the Virtex-6 ML605 Evaluation Kit. Customers can order both of these new products today.

Przybus described the Spartan-6 FPGA SP601 Evaluation Kit (Figure 2) as a low-cost, entry-level environment for developing consumer, infotainment, video and other cost- and power-sensitive applications using the Spartan-6 LX16 FPGA. The kit's system-level capabilities include DDR2 memory control, flash, Ethernet, general-purpose I/O and UART. The kit include ISE Design Suite 11.2 WebPACK, reference designs, a "getting started" demo, board design files, documentation, cables and a power supply.

Meanwhile, the Virtex-6 FPGA ML605 Evaluation Kit (Figure 3) is a scalable environment for developing system designs with the Virtex-6 LX240T FPGA. Among other system-level capabilities, the kit includes high-speed serial transceivers, PCIe® Gen2 blocks, a soft DDR3 memory controller, Gigabit Ethernet and DVI. It also includes the ISE Design Suite 11.2 Logic Edition, reference designs, a "getting started" demo, board design files, documentation, cables and a power supply.

## Three Easy Steps to Kick-Start Your Design

Przybus said that Xilinx designed these kits to get users up and running right out of the box in a three-step process.

In step 1, users connect cables from the board to their PC, power up the board, load the reference design interface software on their computer, view the reference design demo and begin working.

In step 2, users will evaluate the reference design, which includes alternative design implementations using hard and soft IP to implement common functions. Customers can evaluate features directly from the base reference design and see the results visually as well as view key performance statistics.

In step 3, users will open the ISE Design Suite design tools, customize the reference design and generate a new design in the software. Next, they will download it to the evaluation board and then run the design on the FPGA. "It's really that simple," said Przybus. "In a matter of minutes, you can be up and running."

With the release of ISE Design Suite 11.2, the Xilinx tool suite now supports the Virtex-6 and Spartan-6 FPGA families, delivering a 2x overall run-time improvement, speeding synthesis runs by better than 2x using XST and offering place-and-route optimizations that lower overall

| # | Feature |
|---|---|
| 1 | DDR 3 |
| 2 | GPIO Dip Switch |
| 3 | SFP |
| 4 | Flash |
| 5 | 10/100/1000 Ethernet |
| 6 | USB UART & USB JTAG |
| 7 | MGT Clock |
| 8 | User Clocks |
| 9 | FMC Connectors |
| 10 | Pushbuttons |
| 11 | MGT |
| 12 | USB 2.0 |
| 13 | Card Reader for System ACE |
| 14 | DVI Output |
| 15 | PCI Express |
| 16 | 12V Wall Adapter Power |
| 17 | 12V ATX Power |
| 18 | Power Regulator Control |

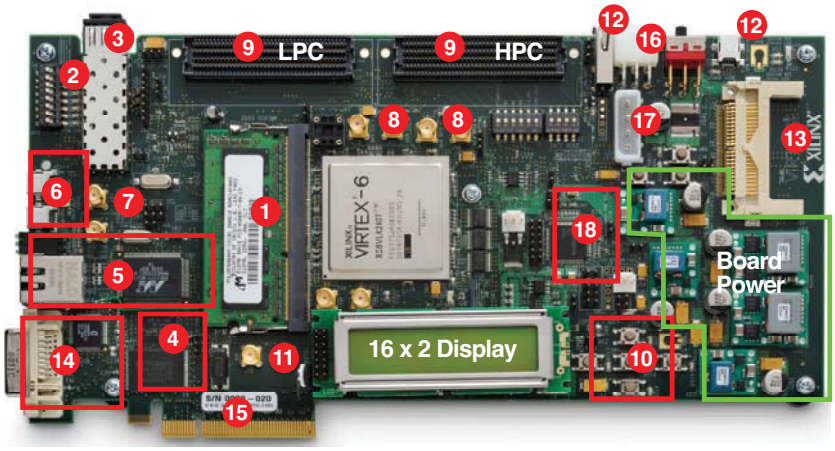



*Figure 3 – The Virtex-6 ML605 Evaluation Kit offers world-class functionality.*

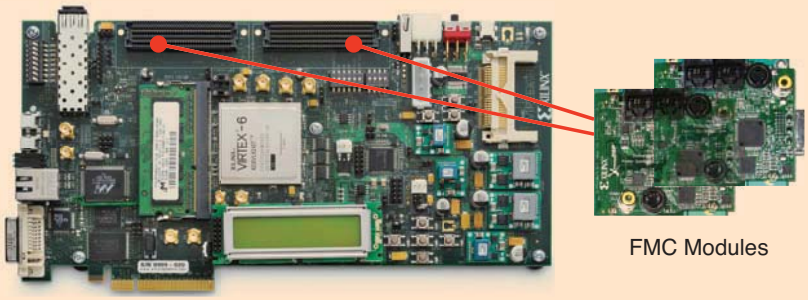
## FMC Card Expedites Design Development

One of the key enablers of the Targeted Design Platform approach is the mutual adoption by Xilinx and its network of resellers and IP partners of the FPGA Mezzanine Card from the VITA standards body. The FMC will serve as a standard interface to attach Domain-Specific and Market-Specific Kits from Xilinx and its partners to the Base Evaluation Kits (see figure).

This standards-based approach allows Xilinx and its network of third-party component and board suppliers, including Avnet Electronics, Curtiss-Wright Controls Embedded Computing, Linear Technology and Northwest Logic, to deliver to mutual customers their latest and greatest offerings.

The ANSI-approved VITA 57.1 standard incorporates predefined and fixed locations of parallel and serial I/Os, clocks, JTAG, control signals and power. It uses the well-defined high-performance Samtec SeaRay, supporting Low Pin Count (LPC) 4x40-row and High Pin Count (HPC) 10x40-row connection. For compatibility with older Xilinx FPGA boards, it also includes a voltage-compatible FMC HPC/LPC module.

*— Mike Santarini*




*The FPGA Mezzanine Card (FMC) connection will facilitate rapid IP and kit development.*

dynamic power consumption by 10 percent. In addition, the tools have a 28 percent smaller workstation memory footprint than the prior version of ISE.

ISE 11.2 supports the SecureIP simulation model, facilitating compatibility with third-party simulators from Cadence, Mentor Graphics and Synopsys. Also, Mentor Graphics' Precision RTL and Precision RTL Plus products support the Base Targeted Design Platform, as do the Synplify Pro and Synplify Premier tools from Synopsys (Synplicity).

Certainly, another key component of the Base Targeted Design Platform and the overall Targeted Design Platform strategy is IP support. In conjunction with the release of the Base Targeted Design Platform, Xilinx and its IP partners are rolling out numerous soft cores supporting the Spartan-6 and Virtex-6 FPGA families.

The upcoming domain-specific and market-specific offerings will feature some of these pieces of IP. One, for example, is a PCI Express® DMA Engine from Northwest Logic. Xilinx and Northwest will package the DMA Engine with a demonstration application and drivers to form a complete Connectivity Targeted Reference Design built to support the new Spartan-6 and Virtex-6 devices with the Xilinx base platform.

Xilinx is selling the Spartan-6 FPGA SP601 Evaluation Kit for $295 and the Virtex-6 FPGA ML605 Evaluation Kit for $1,995. The Spartan-6 FPGA SP601 Evaluation Kit is available now. The Virtex-6 FPGA ML605 Evaluation Kit will be available in late July.

In the third quarter, Xilinx and reseller Avnet will begin rolling out domain-specific kits for the connectivity, embedded and DSP spaces, followed by market-specific kits for communications, video and broadcast.

For more information, contact your local Xilinx sales office or distributor.

# Quickly see inside your FPGA

Get the most out of your test equipment and shave time out of your next debug cycle with Agilent oscilloscopes and logic analyzers. Our innovative FPGA dynamic probe application allows you to quickly connect to multiple banks of internal signals for rapid validation with real time measurements. Reduce errors with automatic transfer of signal names from the .cdc file from your Xilinx Core Inserter.

Toggle among banks of internal signals for incremental real-time internal measurements without:

- Stopping the FPGA
- Changing the design
- Modifying design timing

Shown with: 9000 Series oscilloscope and 16900 logic analyzer with FPGA dynamic probes

*Also works with all InfiniiVision and Infiniium MSO models.*

**See how you can save time by downloading our free application note.**

**www.agilent.com/find/fpga_app_note**



**u.s. 1-800-829-4444 canada 1-877-894-4414**

Agilent Technologies

# Blade Management Controller Rides FPGA Embedded Processor

## CloudShield builds powerful, flexible solution around Xilinx PowerPC 440 running embedded Linux.

by Andy Norton
Distinguished Engineer, Office of the CTO
CloudShield Technologies, Inc.
*ANorton@CloudShield.com*

Jeff Mullins
Principal Engineer, Embedded Systems Lead
CloudShield Technologies, Inc.
*JMullins@CloudShield.com*

Dick Mincher
Embedded Systems Specialist
CloudShield Technologies, Inc.
*DMincher@CloudShield.com*

The trend toward converged networks for both telecom and enterprise simplifies network infrastructure and dramatically lowers costs while providing scalable, open platforms. Blade platforms are achieving network convergence using Ethernet, scaling from 1G/10G to 40G/100G, with blade-management controllers keeping pace by using standards-based Intelligent Platform Management Interfaces (IPMI).

To help drive network convergence into the mainstream market, our design team here at CloudShield Technologies architected a flexible blade-management controller by integrating the PowerPC® 440 found in the Virtex-®5 with standard and custom peripheral cores, creating multiple embedded-system configurations with a common hardware design. This FPGA-based design took advantage of a unique and flexible feature set that included system reconfiguration, powerful embedded processors, intellectual-property (IP) cores and embedded Linux firmware for chassis management services unique to the resident blade.

Xilinx Platform Studio cores allowed us to quickly instantiate UARTs, I²C buses, memory interfaces, Ethernet MAC/PHY combinations and a system monitor for voltage and temperature oversight. We easily created custom cores to provide novel system acceleration, offloading real-time tasks. Our application implemented standards-based IPMI and chassis management firmware running over embedded Linux, for a robust and open software infrastructure.

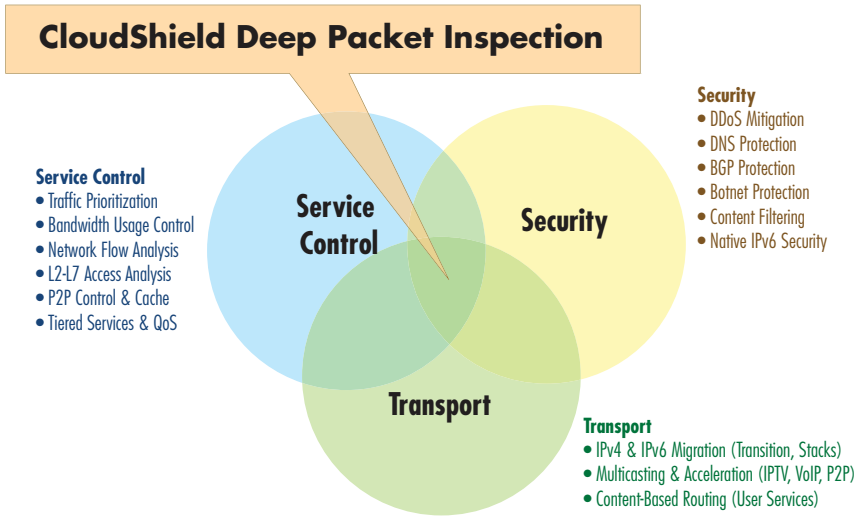



*Figure 1 — Next-generation deep-packet inspection processing system enables convergence of service control, security and transport functions into integrated services and capabilities.*

## Deep-Packet Processing

Our next-generation content-processing platform is a highly programmable, modular, high-speed hardware system capable of handling a wide range of network applications and multiple simultaneous functions, as shown in Figure 1. The chassis integrates deep-packet processor blades, application-services processor blades, redundant chassis management modules and an array of interface modules that can be deployed with different network interface options and configurations.

Chassis and blade management are under the control of redundant chassis management modules using a variant of the Intelligent Chassis Management Bus (ICMB) that extends IPMI intelligent platform management to meet highly secure requirements. Each blade or module of this bus—except for the power supply module—contains an IPMI management controller to share environmental monitoring and status with the chassis management module.

Sharing common requirements, our family of management controllers consists of chassis management controllers (CMCs), processor management controllers (PMCs) on processor blades and interface management controllers (IMCs) on interface modules, as shown in Figure 2.

The chassis' extensive networking features support four control-plane Gigabit Ethernet links to each blade. The data plane has 16 high-speed lanes to each blade that provide four 10-Gbit Ethernet links using XAUI (4 x 3.125 Gbits/second). Future bandwidth scalability exists, with each lane capable of 10GBASE-KR (single lane, 10.3125 Gbps). We used Xilinx® Virtex-5 FPGAs to implement control-plane Gigabit Ethernet functions with hard trimode Ethernet media-access controllers (TEMACs) as well as data plane 10-Gbit Ethernet using the Xilinx 10GEMAC IP core with integrated XAUI (using RocketIO™ serial transceivers).

## Blade Management Services

Our chassis management solution is a hierarchical structure, with the IMC and PMC entities operating in much the same way as a standard IPMI baseboard management controller (BMC) for their respective module or blade. For each blade, the controllers use IPMI commands to monitor voltage and temperature, provide a power control interface, enable blade network interface configuration and allow interrogation of manufacturing data.

As the center of chassis management, the CMC shares common IPMI BMC board-level features with the IMC and PMC, along with additional higher-level chassis management functionality. The dual redundant CMCs operate in an active/standby capacity with automatic health monitoring and failover. This chassis management capability allows the CMC to discover, inventory and provide power permissions for all blades and modules. To prevent oversubscription of the available power supplies, the active CMC regulates power demand for the entire chassis by allowing the IMCs and PMCs to power on only when capacity remains. The fan tray and power supply modules use a chassis I²C bus where the active CMC ensures sufficient power and cooling for the entire chassis.

Once the CMC detects a new blade in the chassis, it uses a discovery protocol to locate and identify each blade. The discovery protocol exchanges IPMI messages to retrieve serial number, model number and inventory for the blade or module, and then sends a command with power permissions if the blade can be correctly identified and power is available. The CMC monitors events from each IMC or PMC to detect warning and critical conditions, collecting and storing them in an event log.

To provide a rich user feature set, our primary user interface to the chassis management system is through a window management or command line interface (WMI or CLI) running on an application processor blade. The application processor blade either occupies a chassis slot or remotely communicates through a control-plane network connection to the chassis management controller. The CMC autonomously provides chassis oversight for health and

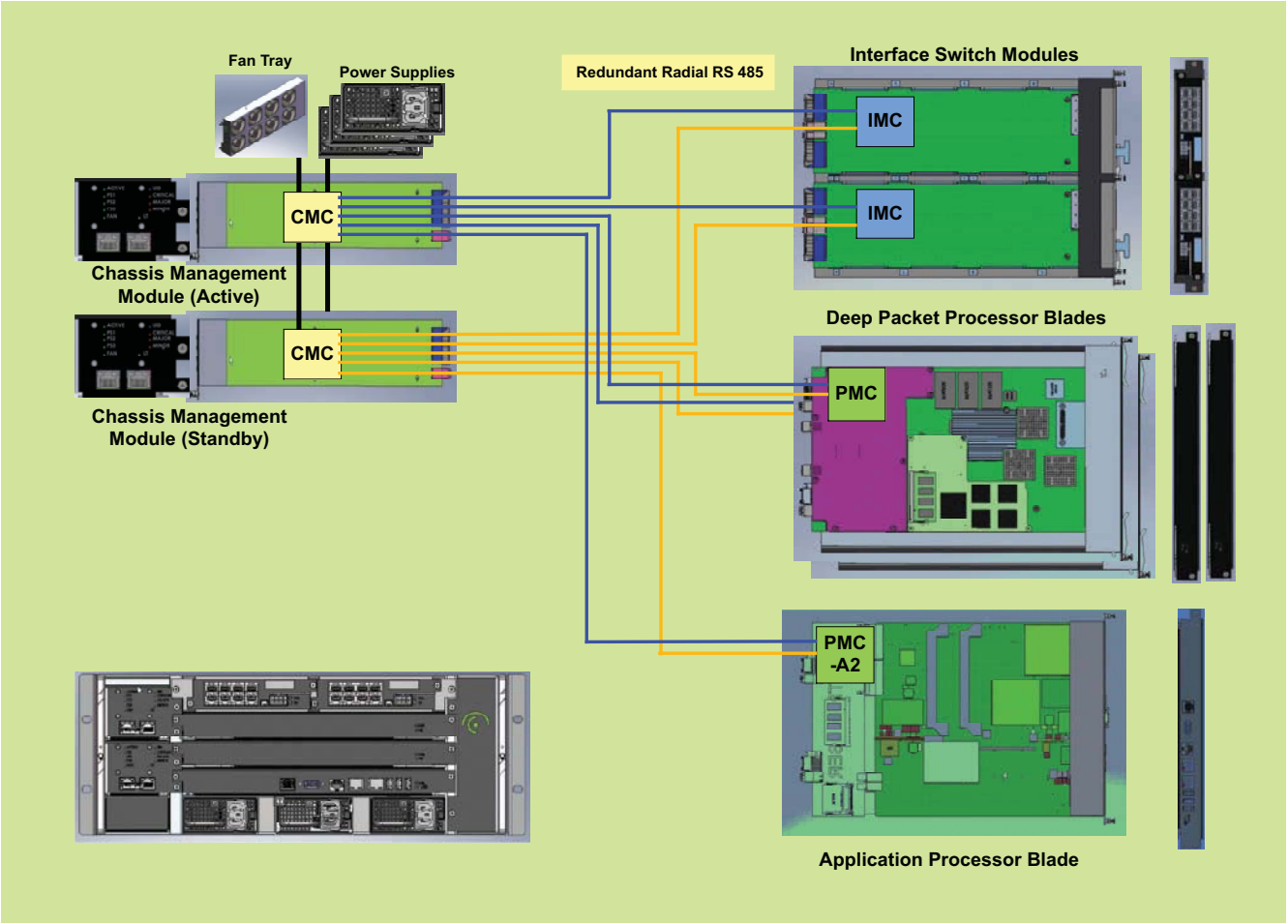



*Figure 2 – Secure ICMB chassis management system*

environmental monitoring, presenting this information through front-panel LEDs, while application-specific user control comes from the application processor blade. This same blade interrogates all blades and modules, including the CMC, to provide the user with environmental as well as operational health and status. The CMC contains an RS485 IPMI proxy service to pass application processor blade commands on the control plane through to the RS485 IPMI path to the target blade.

Each CMC has five independent point-to-point RS485 links to each blade. Unlike a bus topology, this architecture provides increased security from eavesdropping, minimizes contention and increases reliability, in that a failing blade cannot disrupt communication to other blades. Our redundancy implementation requires the standby CMC to constantly monitor the health of the active CMC. Upon detection of a failure, the standby CMC can take over, with its independent set of radial RS485 links, to keep the chassis operational.

## Management Controller Hardware Architecture

Sharing common hardware functionality, our controllers include FPGA multiboot and fallback capability, PowerPC440 processor, SDRAM and NOR flash memory configuration, interrupt controller, serial console UART, ICMB RS485 custom core, Gigabit Ethernet links, system monitor sensors and local $I^2C$ buses.

Our controller family is uniquely architected to share a common hardware platform using a single microprocessor hardware specification file that defines the embedded-processor core instantiations, connectivity and parameterization. Our mini boot loader resides in internal block random-access memory (BRAM), which copies the U-Boot boot loader from flash to DDR memory and then jumps to the U-Boot entry point.

Taking advantage of the features of the Virtex-5 FX30T FPGA, CloudShield's team easily assembled an advanced embedded system, integrating the PowerPC 440, DDR2 memory controller, multiport memory controller for flash, Ethernet MAC/PHYs, system monitor and other standard cores with our custom core.

By leveraging the Virtex-5's multiboot reconfiguration, the system achieves true in-system FPGA upgradability without a processor to manage bitstream loading. Our external flash supports both "golden" and "upgrade" bit files located at addresses the revision select pins determine. The FPGA's ability to fall back and reconfigure using the golden image in the event of an upgrade image failure was essential to our design objective of a configurable high-availability system.

Post-configuration, the PowerPC 440 core operates at 400 MHz, the crossbar and MPLB bus at 133 MHz, TEMACs at 125 MHz and DDR2 at 266 MHz. Our constructed hardware system covered the superset of functionality needed among the CMC, IMC and PMC variants as shown in Figure 3. The flash contains enough space for multiple application programs, including the CMC, IMC and PMC configurations.

We were able to rapidly craft the base system thanks to the wide variety of standard peripheral cores available in the IP library. We selected cores as complex as Gigabit Ethernet (with choices of 1000BASEX, SGMII and GMII PHY interfaces) and as simple as traditional 16550 UARTs to meet our system requirements.

Key requirements for a blade controller involved chip- and board-level monitoring and alarms for voltages and temperatures. We took advantage of an integrated system monitor, simply instantiating the SYSMON_ADC pcore from the standard IP library. The system monitor is a hard macro on Virtex-5 devices that contains a 10-bit, 200-kilosample/s analog-to-digital converter, supporting both on-chip and off-chip monitoring of supply voltages and temperatures.

Off-the-shelf IP and the availability of an Avnet Virtex-5 FXT evaluation kit that used the XC5VFX30T FPGA allowed us to rapidly prototype our design and kick-start the software development effort right out of the box.

With the software effort under way, we then created custom IP cores to satisfy a number of needs. While off-the-shelf pcores might meet functional requirements, multiple instances of simple pcores such as GPIO can result in excessive Processor Local Bus (PLB) loading as well as higher slice utilization. Rather than incur the expense of multiple IP interfaces, we built a single super-GPIO module with the help of the Create Import Peripheral (CIP) wizard integrated into XPS. Amortizing a single IP interface over multiple GPIO registers, we collapsed large numbers of internal and external registers into a single custom pcore including revision, scratch, and internal and external GPIO registers.

We also used the CIP wizard to build our special-purpose custom IP. The wizard created the necessary files (MPD, PAO, HDL templates) and directory structure, and offered easy selection of IP interface services and user logic interfaces. It also generated the bus functional model needed for simulation. Our novel PicoBlaze™-based coprocessor pcore is described later in greater detail.

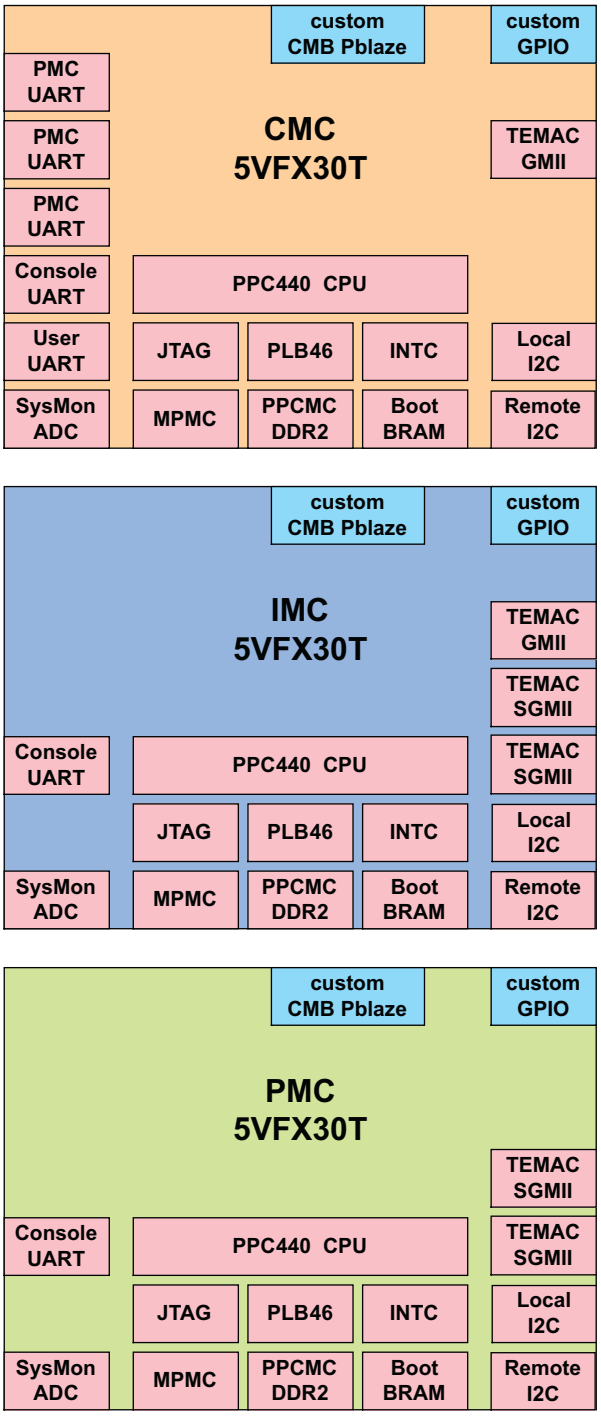



*Figure 3 – Common hardware architecture for CMC (top), IMC (center) and PMC versions.*

The embedded Linux operating system could not natively guarantee interrupt latency. Our solution leveraged the capabilities of the PicoBlaze 8-bit sequencer.

### IPMI Messaging

Our chassis management controller with its corresponding system interfaces is shown in Figure 4. We use the custom ICMB coprocessor pcore for IPMI messaging to the processor-management and interface-management blade controllers. GPIOs monitor and control the front panel, backplane, redundancy and interlocks, as well as internal registers, while TEMACs are used for control-plane Ethernet switch management. Our $I^2Cs$ read the local module EEPROM, and handle remote monitoring of chassis fans and power supplies. UARTs take care of front-panel user connectivity, debug consoles and proxy UARTs to CPUs on processor blades.

The real-time protocol requirements of the Intelligent Chassis Management Bus presented us with a problem in that the embedded Linux operating system could not natively guarantee interrupt latency.

Our solution uses a custom ICMB pcore with the PicoBlaze 8-bit sequencer as an ICMB data-link and physical-layer coprocessor offloading the PowerPC. This underappreciated resource (see *Xcell Journal*, issue 67, page 53, "A Hidden Gem: PicoBlaze IP") is shown in Figure 5, coupled with BRAM instruction store, registers, communication FIFOs, UARTs and timers. After initialization, the PowerPC holds the PicoBlaze in reset, loads code into the PicoBlaze program BRAM and then removes reset, activating the coprocessor.

We wrote our PicoBlaze code in the form of a state machine to monitor the transmit FIFO and control registers, implement the collision-avoidance and back-off algorithms, and manage five UARTs.

The TX FIFO passes 8-bit data from the PowerPC to the PicoBlaze for transmission on the ICMB bus. To transmit a packet, the PowerPC writes the packet into the TX

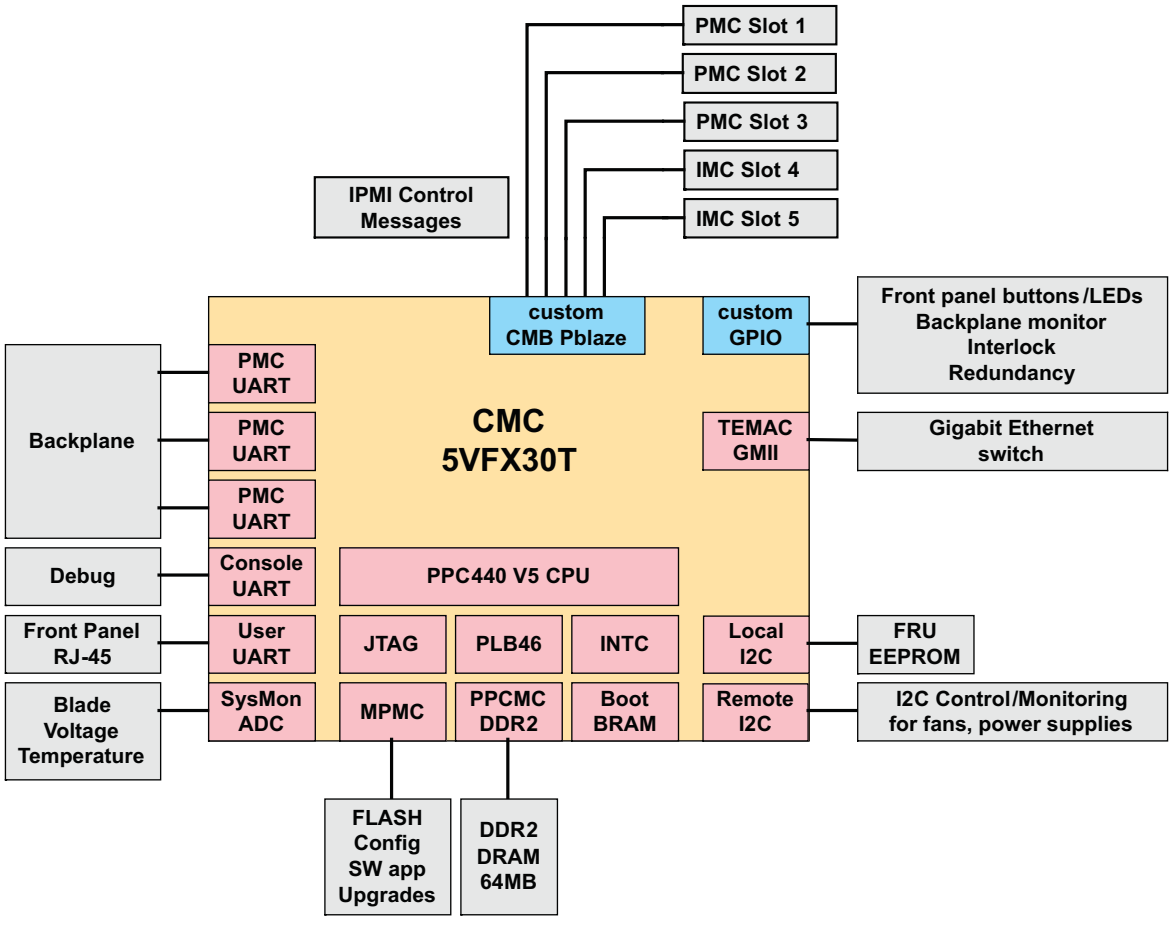



*Figure 4 – CMC system interfaces*

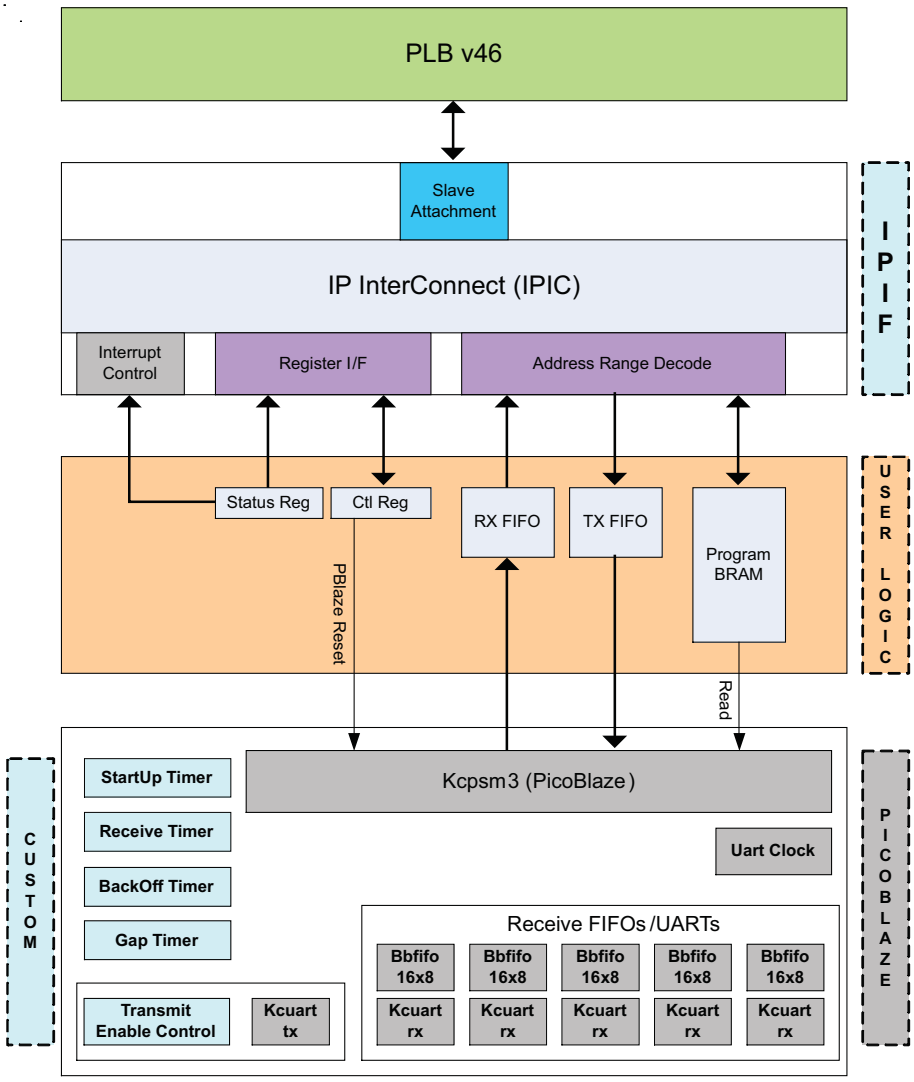



*Figure 5 – Custom ICMB PicoBlaze pcore*

FIFO, followed by a write to the control register with a "start" bit and channel number. Using a common transmit UART and individual transmit enables, the PicoBlaze enables a single RS-485 transceiver for data transmission. Upon completion, the PicoBlaze indicates success or failure by writing to the status register and generating an interrupt to the PowerPC.

Packets come in on multiple UARTs simultaneously. The PicoBlaze maintains individual state information for each channel. Indications of start/end of packet and data bytes are placed in the RX FIFO along with the channel number, and the PowerPC gets an interrupt when data is available. The RX FIFO passes 8-bit data, 5-bit status (start/end of packet, error indications, debug information) and 3-bit channel number information from the PicoBlaze to the PowerPC. The PowerPC then demultiplexes packet data and, when valid, sends it up to the servicing agent. We included additional trace and state transition capability in the receive channel in order for the PowerPC to print debug messages to the console.

We used the CIP wizard to create a bus functional model, enabling us to generate bus stimulus and verify the operation of the peripheral core without needing to create an extensive test bench or full-system simulation. The wizard made it easy to generate, respond to and analyze bus transactions. As a result, we created and modified our simulation for the custom pcore and got it running in a matter of hours.

We used the output of the PicoBlaze assembler to initialize the program BRAM. Next, using the Bus Functional Language to describe PLB bus transactions, we generated a PicoBlaze reset, sent data to the TX FIFO and wrote to the control register to transmit data on a channel. Looping back transmit and receive ports was a simple way to verify the operation of the receive channels and RX FIFO.

### Firmware Architecture

There are many embedded-OS choices available for the PowerPC 440 and, specifically, the Xilinx hard PPC440 core. All of the management controllers required real-time performance, a robust network stack and extensibility for our custom peripherals. Our choice of embedded Linux allowed us to take advantage of existing open-source resources, utilities, optimizations and support.

After reset, the PowerPC boot sequence begins from reset vector space within a small internal BRAM. Mapped into this area is a small mini boot loader (12 kbytes), which looks for a valid U-Boot image at an "upgrade" or "golden" location within the NOR flash. Since the golden image is programmed only at the factory during the manufacturing process, this choice eliminates the possibility of a failure during flash programming of a new U-Boot version. Once it detects and validates the proper U-Boot image, the PowerPC loads it into SDRAM and executes it. U-Boot then loads the proper Linux image files and passes control for the final time. At this point, a fully configured Linux kernel loads and begins our application-specific processes.

We considered several options for the boot sequence during the design process. For example, the mini boot loader could have mapped the NOR flash into the reset vector area and eliminated the BRAM. This would have saved some BRAM, but the processor would then be dependent on a valid NOR flash part and image, which could be corrupted during a

failed flash attempt. Similarly, a case could be made for eliminating U-Boot by loading and executing Linux directly. While this could be a much more direct boot sequence, we found the features of U-Boot during the development phase to be extremely valuable for prototype bring-up and debugging.

Our initial design challenge was to locate and choose the correct open-source trees. DENX Software Engineering (*www.denx.de*) has extensively developed the Das U-Boot boot loader for a variety of embedded-processor boards. But while it directly supports the MicroBlaze, PowerPC 405 and PowerPC 440 processor cores, the DENX version does not include the latest patches, board support or drivers for some of the XPS IP cores that come with the EDK.

Therefore, we chose to use the Xilinx development branch of the U-Boot tree (*xilinx.wikidot.com*), which contains the latest changes and the drivers for the I$^2$C, ENET and LLTEMAC, providing additional functionality without starting from scratch. Xilinx frequently updates its development branch from the original DENX master tree, submitting changes upstream for acceptance into the DENX tree.

Having selected the Xilinx U-Boot tree, we looked at the main Linux tree (*www.kernel.org*) and its support for the Xilinx drivers. We similarly concluded that we obtained more working functionality when choosing the Xilinx-maintained Linux development branch (also at *xilinx.wikidot.com*), which included the latest changes that may not necessarily be in the main Linux tree.

A potential drawback of using the Xilinx-maintained trees is that there is lag time in terms of synchronization with the main trees, and they are officially development-quality branches. Nevertheless, we selected the Xilinx trees to support the LLTEMAC cores and newer features in our design.

In order to build U-Boot and Linux, we needed a Linux distribution and machine to run the scripts and tools that cross-compile, link and build the PowerPC executable. While this could be a barrier if only Windows-based machines were available, there are now a few Windows virtual machines that run Linux within the Windows environment. The added benefit is that once we set up a build machine, we could back up the virtual-machine disk image or replicate it for use on other development machines. We selected Sun xVM VirtualBox as a lightweight, reliable and easy-to-use virtual machine that will run many different OS types, including Linux.

While the Linux kernel provided an excellent foundation, the actual drivers and applications handled the entire workload specific to our application. The Xilinx I$^2$C and LLTEMAC drivers connect the Linux IP stack to the hardware, and the CloudShield ICMB driver hooks the custom PicoBlaze IP core into the Linux driver subsystem. We used Busybox as a standard Linux application, and added three new CloudShield applications for chassis management by the CMC, as shown in Figure 6.

Our RS485 proxy application connects to a standard IP port and communicates over the control-plane Ethernet, decoding each packet. Packets received from our WMI/CLI on the application processor blade may be addressed to three different targets in the system: BMC functions on the chassis management module board, CMM blade management for the chassis or other blades within the chassis. All baseboard management controller functionality runs over the I$^2$C buses, handling CMC and chassis health requests. The RS485 proxy application translates packets destined for other blades in the chassis into ICMB packets and sends them over the RS485 link to the appropriate destination. Blade management packets allow the user to obtain chassis status, such as power subscription, as viewed by the CMC. Some additional minor applications for redundancy and front-panel serial console interaction complete the CMC.

Our IMC and PMC were very similar to the CMC firmware, except that we replaced the blade-management application with a switch- or processor-management application appropriate for that blade.

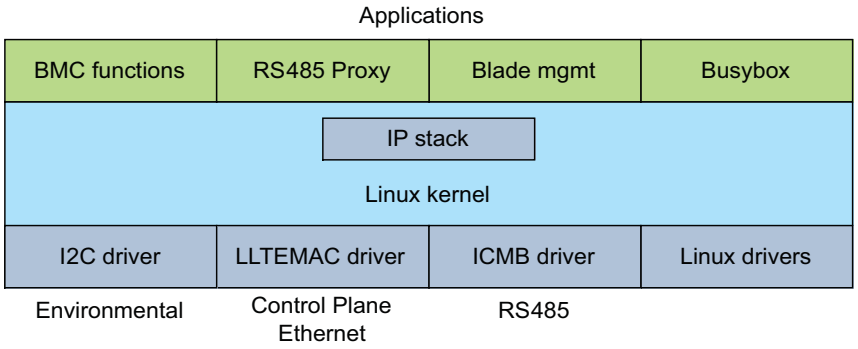


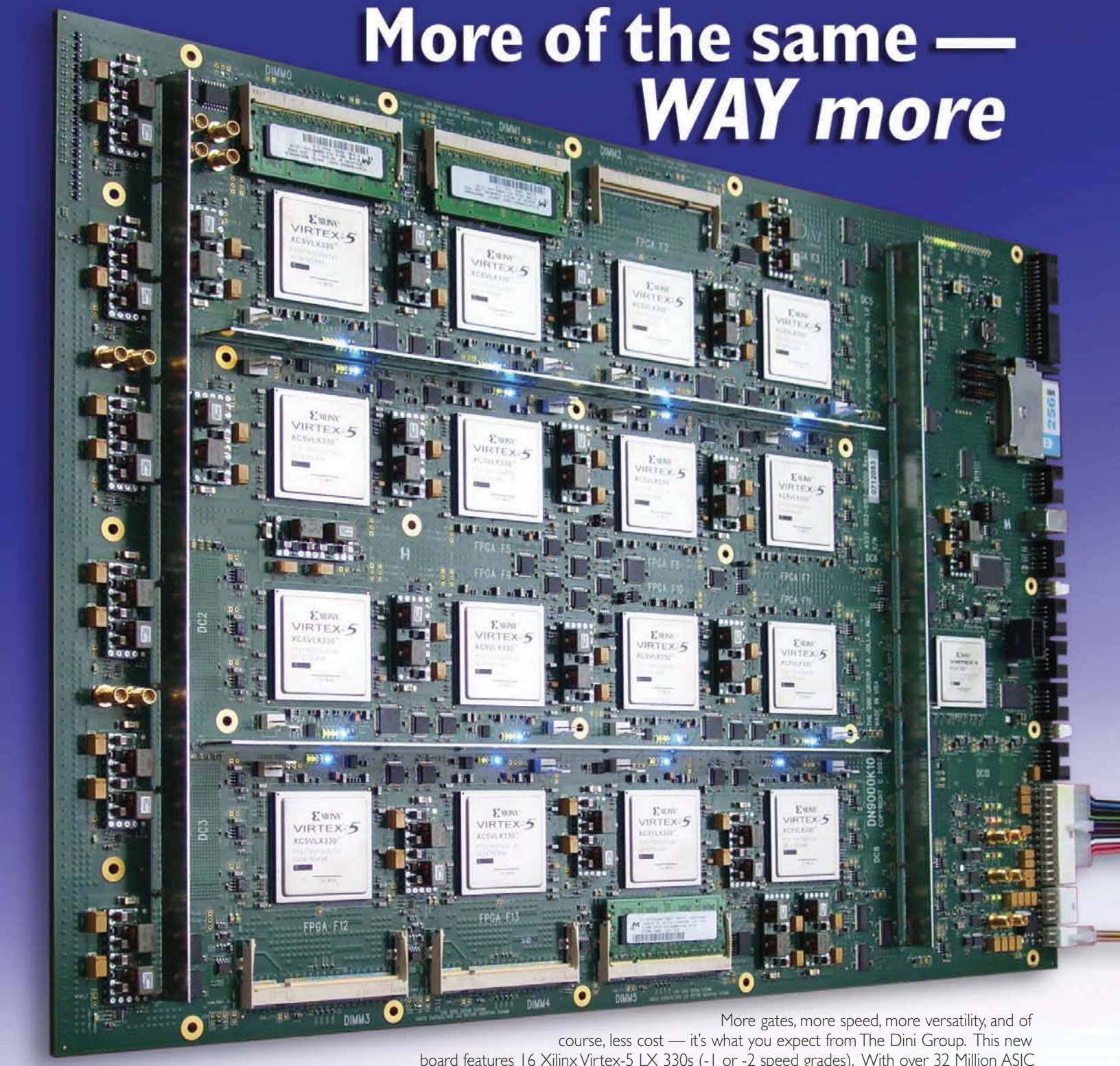

*Figure 6 – CMC firmware architecture*

## Architect, Build, Verify, Deliver

The key to rapidly architecting, developing and delivering complex embedded-system solutions is "right-sizing" the choices, efforts and methodology. The combination of powerful embedded processors in FPGAs, off-the-shelf IP and accelerated design and verification of custom IP enabled us to create multiple embedded-system configurations with a common hardware design. The benefits of using Linux and an integrated development environment freed us to focus on the target application while utilizing the available open-source services, drivers and libraries.

Indeed, the ability to rapidly alter internal FPGA embedded architecture, easily deploy in-system upgrades and exploit new hardware-software trade-off boundaries is transforming embedded-system design.

# More of the same — *WAY more*




More gates, more speed, more versatility, and of course, less cost — it's what you expect from The Dini Group. This new board features 16 Xilinx Virtex-5 LX 330s (-1 or -2 speed grades). With over 32 Million ASIC gates (not counting memories or multipliers) the DN9000K10 is the biggest, fastest, ASIC prototyping platform in production.

User-friendly features include:

- 9 clock networks, balanced and distributed to all FPGAs
- 6 DDR2 SODIMM modules with options for FLASH, SSRAM, QDR SSRAM, Mictor(s), DDR3, RLDRAM, and other memories
- USB and multiple RS 232 ports for user interface
- 1500 I/O pins for the most demanding expansion requirements

Software for board operation includes reference designs to get you up and running quickly. The board is available "off-the-shelf" with lead times of 2-3 weeks. For more gates and more speed, call The Dini Group and get your product to market faster.

The DiNI Group

**www.dinigroup.com • 1010 Pearl Street, Suite 6 • La Jolla, CA 92037 • (858) 454-3419 • e-mail: sales@dinigroup.com**

# Virtex-5 Propels Ultrawideband Comms and Ranging

## Europe's PULSERS project uses Xilinx FPGA with MicroBlaze processor in impulse-radio UWB system.


by Guy Eschemann
Research Engineer
Sennheiser electronic GmbH & Co. KG
*Guy.Eschemann@gmail.com*

Heinz Lüdiger
Project Manager
IMST GmbH
*luediger@imst.de*

Birgit Kull
Senior Scientist
IMST GmbH
*kull@imst.de*

Since the Federal Communications Commission authorized the unlicensed use of the ultrawideband (UWB) radio technique in 2002, most of the technology's commercial applications, such as wireless USB, have been based on frequency-domain modulation techniques (such as OFDM) for high-data-rate transmissions. Alongside this established approach, UWB offers the potential for another kind of data transmission based on ultrashort pulses with durations on the order of nanoseconds. So-called

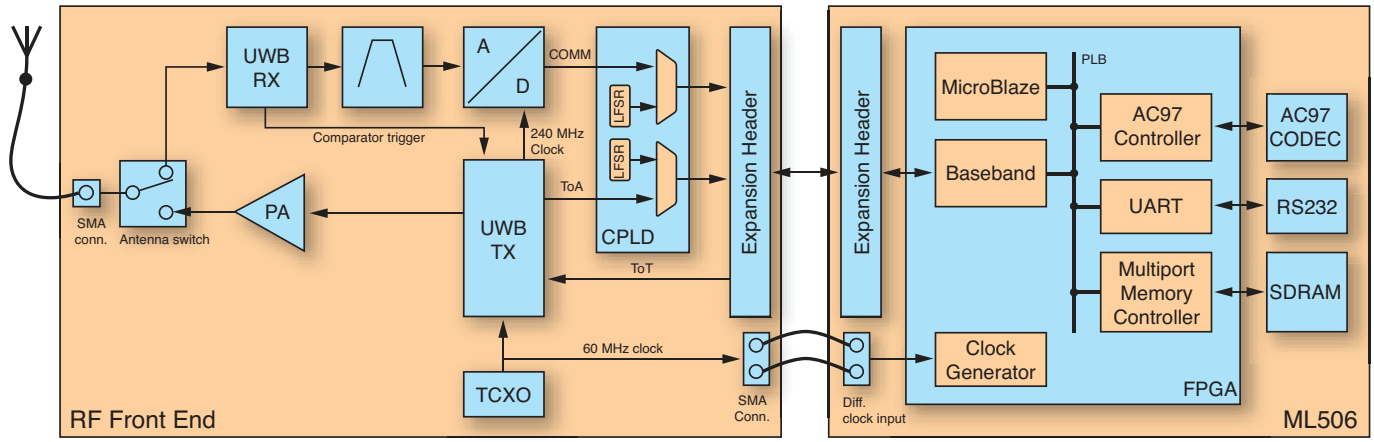



*Figure 1 – The system consists of an off-the-shelf Xilinx ML506 board connected to a custom UWB daughterboard.*

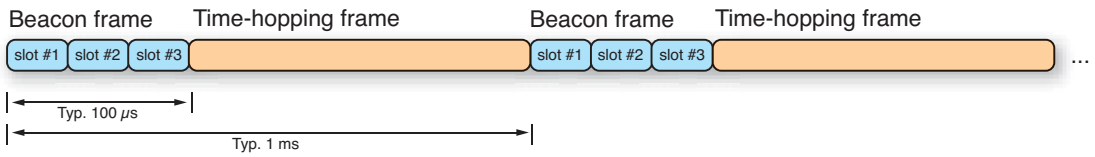



*Figure 2 – Periodic beacon frames consisting of three beacon slots are interspersed with time-hopping frames.*

impulse-radio (IR) systems transmit information by modulating one or more of the pulse parameters, such as position or amplitude. At the same time, by measuring the pulses' time-of-flight, they can implement centimeter-accurate ranging capabilities.[1] This opens up the field to a new range of location-aware applications in such diverse areas as logistics (package tracking), manufacturing, search and rescue (communication with and localization of firefighters, for example) or smart tour guides.

The European PULSERS Phase II Project, an industry-led collaboration into UWB radio technology by 30 key industrial and academic organizations, has set out to design and implement an IR-UWB communication and ranging system [2] featuring data transmission capabilities in the megabit/second vicinity and a ranging resolution of 4 cm. Our system consists of a set of identical autonomous nodes, each of which can communicate with and determine the range to every other node in the network. A node consists of a custom UWB daughterboard connected to an off-the-shelf Xilinx® ML506 development board (see Figure 1). The high performance of the Virtex®-5 SXT architecture combined with the flexibility of a MicroBlaze™ soft processor allowed us to implement the entire baseband signal chain as well as all the higher system layers in a single FPGA.

## IR-UWB Communication and Ranging

To transmit information, our system uses a simple pulse-position modulation with four possible time shifts (4-PPM), where each pulse encodes two data bits. Pulses are grouped into frames and transmitted in a predefined raster of beacon frames and time-hopping frames, as illustrated in Figure 2. Each beacon frame consists of three identical beacon slots that customers can use for ranging or communication purposes. We initially intended the time-hopping frame to carry high-data-rate transmissions based on time-hopping code, but we'll use that technique in a later product. At this time, all the data transmissions occur in the beacon frames.

We performed ranging using a method known as two-way ranging, which consists of measuring the time delay between sending a ranging request and receiving the answer from the remote node (see sidebar). Ranging requests are always sent in beacon slot 1, while ranging answers are expected to come back in slot 3. This gives the remote node the duration of a full beacon slot (slot 2, approximately 33 microseconds) for processing the received ranging request and scheduling the outgoing ranging answer.

| | |
|---|---|
| Center frequency | 7.68 GHz |
| Baseband bandwidth | 750 MHz |
| Expected range | 25 - 30 meters |
| Actual range | 3 m (due to local-oscillator crosstalk) |
| Ranging resolution | 3.9 cm |
| Communication data rate | > 1 Mbit/s |

*Table 1 – UWB communication and ranging system characteristics*

## System Architecture

The ultrawideband daughterboard carries both the impulsive-transmitter and incoherent-receiver ASICs, which we designed specifically for this project using IHP's 0.25-micron SiGe:C BiCMOS technology. [3, 4]

The transmitter ASIC, which generates UWB pulses as shown in Figure 3, is capable of modulating both the amplitude and the position of the generated pulses. It includes a 3.84-GHz counter for precisely scheduling the time of transmit of the outgoing pulses and measuring the time of arrival of the received pulses.

The reception path splits into two branches inside the receiver ASIC. The first branch, which has a relatively narrow bandwidth (120 MHz), serves communication and coarse pulse-timing purposes. Precise pulse timing occurs on a second reception branch that utilizes the full impulse bandwidth (750 MHz). On this branch, a high-speed comparator detects incoming pulses. Its output triggers the readout of the 3.84-GHz counter running inside the transmitter ASIC. Thus, the time of arrival of each received pulse is measured with a resolution of 260 ps, which translates into a spatial resolution of approximately 8 cm.

The daughterboard feeds the baseband module in the Virtex-5 FPGA with two data buses running at 120 MHz. The communication (COMM) bus carries the ADC samples while the time-of-arrival bus conveys the high-resolution time stamps associated with the received pulses. Both buses go through an XC95144XV CPLD which, while not strictly required, was a great debugging aid. We can set the CPLD to output a sequence of pseudo-random numbers [5] on the buses to the FPGA. We then use the CPLD output to both adjust the FPGA's input timing and to verify the integrity of the bus lines—tasks that would have been difficult without a priori knowledge of the transmitted data sequence.

Inside the FPGA, the baseband module (see Figure 4) takes care of both the encoding of the outgoing pulses and the decoding of the received pulses. The transmission part of the baseband module is relatively straightforward, consisting mostly of outer (CRC) and inner (convolutional) coding. The implementation of the reception counterpart involves, among other things, a channel estimator and a custom Viterbi decoder, and is thus much more resource-intensive. We connected the baseband module to the processor system via a Processor Local Bus (PLB) interface.

While programmable logic is notoriously more difficult to debug than software, the ChipScope™ Pro tool, with both an integrated logic analyzer and a bus analyzer configuration, was of great help during debugging sessions. The logic analyzer proved useful for simultaneously capturing a burst of COMM and time-of-arrival samples, providing real-world data to our MATLAB® simulator. The bus analyzer helped us debug a few issues related to the PLB interface of our baseband module.

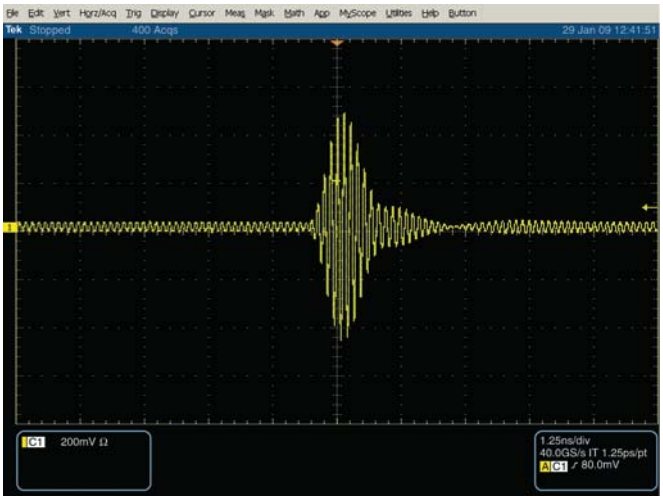



*Figure 3 – A UWB pulse consists of a 7.68-GHz carrier wave with a Gaussian envelope.*

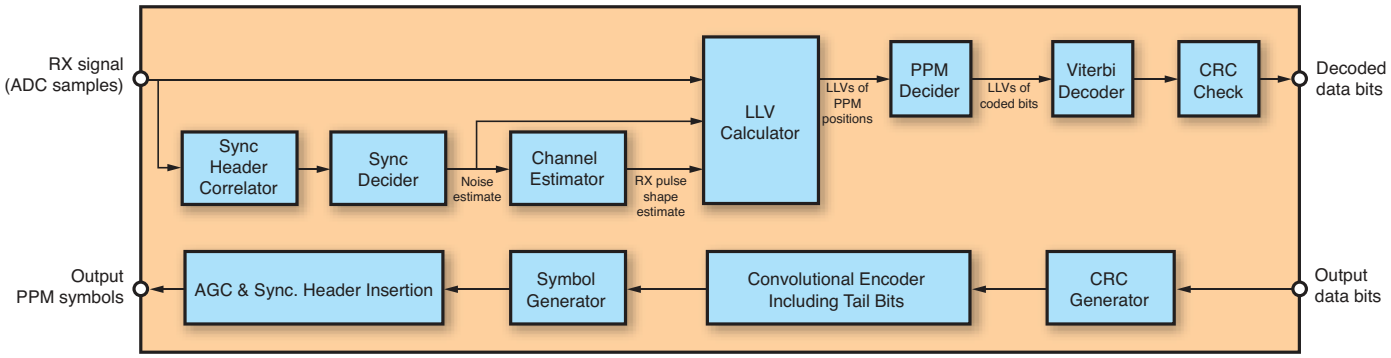



*Figure 4 – The baseband module's reception (top) and transmission chains.*

### Processor System

We generated the processor system using the Base System Builder wizard within the Xilinx Platform Studio (XPS) design tool, which gave us a perfectly working system to start with. After that, we incrementally modified the base system to obtain the system shown in the FPGA section of Figure 1. These modifications involved, among other things, switching to a differential clock input and connecting the baseband module to the PLB.

The software application runs on an embedded MicroBlaze processor on top of Xilkernel, which is a minimal real-time operating system perfectly suited for small applications. The application is divided into three threads that run concurrently:

- The UWB thread manages the configuration and operation of the baseband module.
- The application thread is responsible for the audio acquisition and playback activities when the system is used in data transmission mode.
- The RS232 thread communicates with an external PC running the demonstration graphical user interface.

Since the GNU development chain, which XPS uses, is available on a number of other platforms, we could easily compile and test the hardware-independent modules of code on a host PC (for example, using the Cygwin environment) rather than on the embedded target. That made debugging a lot easier. Only the final testing had to be done on the embedded target, and having a source-level debugger like GDB was a real blessing. Xilinx application note XAPP1037 [6] showed us many useful tricks for debugging our software.

A few hardware issues, located in the UWB ASICs, currently limit the nominal range of our system to 3 meters instead of the initially expected 25 to 30 meters. Still, we have been able to demonstrate both the communication and ranging capabilities of the system, which makes the project a great success.

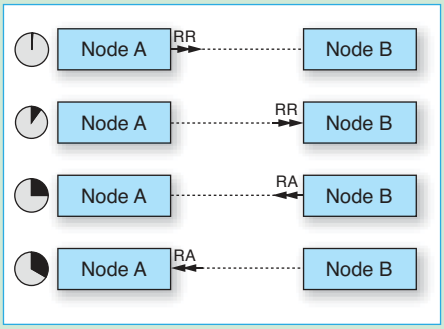
Future work may include redesigning the UWB ASICs to increase the system's operating range, as well as implementing multilateration capabilities to move from a ranging system to an actual indoor positioning system.

For more information, visit *http://www.imst.de/de/forschung_pul.php* or e-mail *luediger@imst.de.*

## Two-way ranging demystified

In two-way ranging, the technology we employed in our impulse-radio UWB system, the distance between node A and node B is determined using the following technique (see figure):

1. Node A sends a ranging request to node B and starts its high-resolution clock (3.84 GHz).
2. Node B receives the ranging request after the signal propagation delay $t_{prop}$, which is proportional to the distance between nodes A and B.




*The two-way ranging process*

3. Node B sends back a ranging answer to node A after a known processing delay $t_{proc}$.
4. Upon receipt of the ranging answer, node A stops its clock at time $t_{rtt}$. It can then compute the one-way signal propagation delay $t_{prop} = \frac{t_{rtt} - t_{proc}}{2}$, which, multiplied by the speed of light, gives the range between A and B.

The 3.84-GHz clock's time resolution of 260 picoseconds yields a spatial resolution of approximately 8 cm. Since, however, the radio signal traverses the distance between the two nodes twice, the range can be determined with a resolution of 4 cm.

Knowing the range between itself and three non-co-linear anchor nodes, a mobile node can compute its 2-D position. Using four non-co-planar anchor nodes, it can even find its 3-D position.

*– Guy Eschemann, Heinz Lüdiger and Birgit Kull*

#### Acknowledgements

*We wish to acknowledge Erwin Stenzel and Daniel Kotzor from EADS; Gunter Fischer from IHP; Thorsten Kohl, Jac Romme and Norbert Schmidt from IMST; Maria Dolores Pérez-Guirao from the Leibniz University of Hannover; Axel Schmidt from Sennheiser, and Dajana Cassioli from RadioLabs for their contributions to this project. Thanks to Steven Backer from Sennheiser for reviewing drafts of this article.*

*This work was co-funded by the European Commission under the Information Society Technologies integrated project PULSERS Phase II.*

#### Citations

*[1] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor and Z. Sahinoglu, "Localization via Ultra-Wideband Radios,"* IEEE Signal Processing Magazine, *July 2005.*

*[2] H. Lüdiger, B. Kull, M. D. Perez-Guirao, "An Ultra-Wideband Approach towards Autonomous Radio Control and Positioning Systems in Manufacturing & Logistics Processes,"* Proceedings of the 4th Workshop on Positioning, Navigation and Communication, *Hannover, Germany, March 2007.*

*[3] G. Fischer, O. Klymenko, D. Martynenko, "Time-of-Arrival Measurement Extension to a Non-Coherent Impulse Radio UWB Transceiver,"* Proceedings of the 5th Workshop on Positioning, Navigation and Communication, *Hannover, Germany, March 2008.*

*[4] O. Klymenko, G. Fischer, D. Martynenko, "A High Band Non-Coherent Impulse Radio UWB Receiver,"* Proceedings of the IEEE International Conference on Ultra-Wideband, *ICUWB 2008, Hannover, Germany, September 2008.*

*[5] P. Alfke, "Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators," XAPP052 application note.*

*[6] B. Hill, "Introduction to Software Debugging on Xilinx MicroBlaze Embedded Platforms," XAPP1037.*

# FPGA-Powered Platform Controls Industrial Motors for Maximum Efficiency

## National Instruments uses Xilinx devices in CompactRIO design for industrial-equipment builders requiring optimal motor control.

by Greg Crouch
Embedded Systems Business Director
National Instruments
*greg.crouch@ni.com*

Facing increased regulation and the need to reduce factory operating costs, machine builders are looking for solutions to boost product power efficiencies. Along with HVAC systems, the top consumers of electricity in a factory are water heating, lighting, office equipment and, especially, machinery. More specifically, the motors within these factory machines are responsible for approximately two-thirds of the total electrical-energy consumption in a typical industrial facility. Motors are everywhere—in blowers, pumps, compressors, conveyors, machine tools, mixers, shredders and more.

One way to get the maximum efficiency from the motors that control machinery is to employ more efficient and sophisticated field-oriented control to optimize their efficiency (see sidebar, "Motor Efficiency Leads to Greener Bottom Line"). To this end, our team at National Instruments (NI) used Xilinx® FPGAs as the basis for a common hardware architecture called reconfigurable I/O (RIO) to create a flexible embedded controller with high computing performance. Our machine-builder customers use RIO as a platform from which to draw field-oriented control (FOC) techniques that improve motor efficiency.

This RIO architecture is now deployed in many systems, such as those from EUROelectronics, Srl. The architecture helped EUROelectronics advance from the prototyping phase to the final machine setup in only three months (Figure 1).

*Figure 1 – Embedded-machine builder EUROelectronics reduced power use with FPGA-based field-oriented control.*

### Reduced Machine Design Time

U.S. Department of Energy data informs machine builders that switching to a motor with a 4 to 6 percent higher efficiency rating can pay for itself in just two years if that motor is in operation for more than 4,000 hours a year. Unfortunately, many machines host motors that are very large and too costly to replace. So in these cases, the key to reaping savings lies in updated drive-control algorithms and controller hardware.

A second challenge is the integrated control complexities required for brushless DC and permanent-magnet synchronous AC motors (PMSM), both commonly grouped together as brushless DC motors (BLDC). Many machine builders lack the software or hardware design expertise required to build an embedded controller that can execute real-time closed-loop control on a wide variety of analog and digital sensor types.

To reduce time to final design for embedded-machine builders, we incorporated one form of our RIO-based architecture into a product called CompactRIO. These FPGA-based configurations include systems built from Xilinx Virtex®-5 LX85 to Spartan®-3 and Virtex-II 1M-gate-based backplanes, combined with PowerPC® 603e-based processors in various performance frequencies (Figure 2).

We built into our RIO framework configuration software utilities and dynamic I/O reconfiguration capabilities that save time in setup and reuse for both the end-

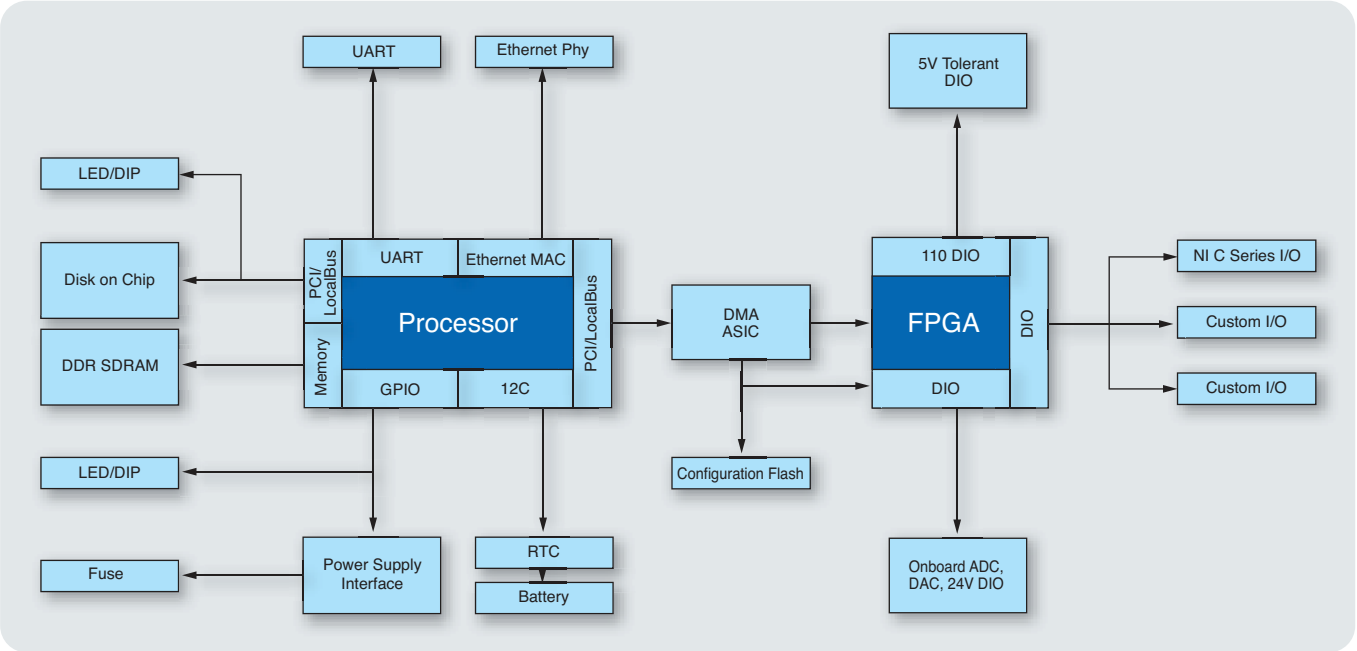



*Figure 2 – Modular RIO architecture-based framework for NI's single-board RIO and CompactRIO configurations*

The machine builder can call both multithread-safe and reentrant functions from multiple threads at the same time. Drivers that lack reentrant execution can erode performance or, worse, cause a crash.

application programmer and the digital design engineer. Our configuration software automatically detects custom hardware installed in the system. Integrated diagnostic tests of I/O peripherals ensure that I/O devices function properly. The designer connects the custom circuitry directly to the Xilinx FPGA, and can debug the code. Our RIO middleware driver architecture includes functionality to integrate simulation code directly into the functional driver, thus simplifying code reuse and debugging.

For example, Figure 3 describes the embedded middleware software design hierarchy. These middleware drivers and

### Help from FPGA Control Algorithms

Although brushless DC motors and permanent-magnet synchronous AC motors are both considered brushless DC motors, they differ in the way their stator is wound. When rotated, the stator of the BLDC is wound in such a way as to produce a trapezoidally shaped back-EMF voltage, while

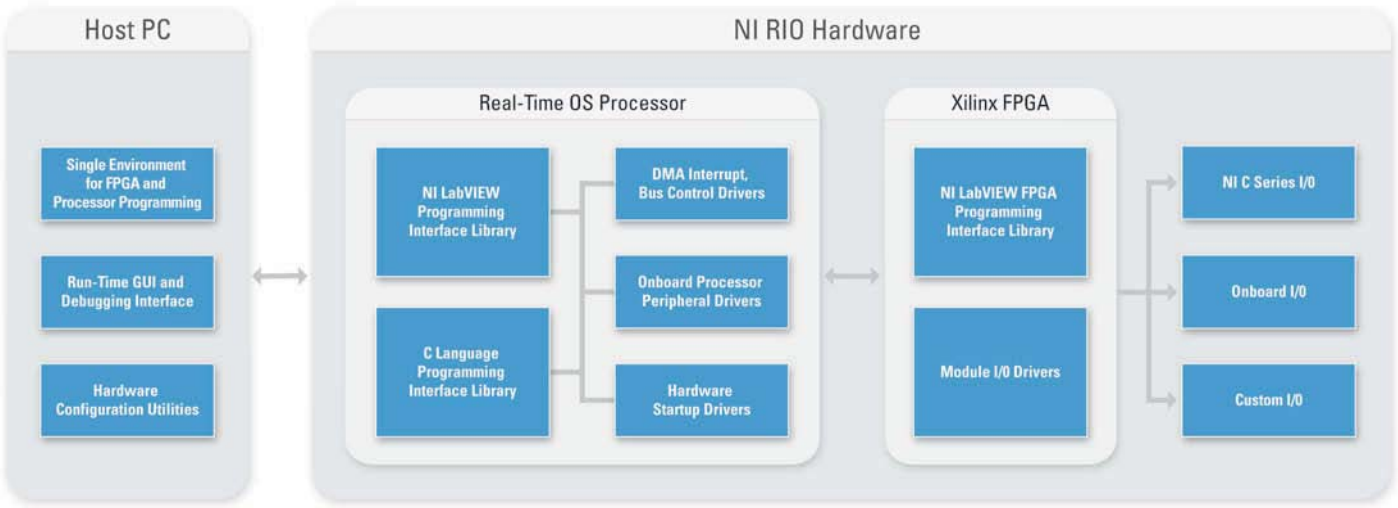



*Figure 3 – With processor middleware drivers, machine builders can focus on custom circuitry design. They can link to the programmable Xilinx FPGAs through standard header connectors.*

design logic with the Xilinx tools or the NI LabVIEW FPGA Module.

Planning for dynamic configuration helps the machine builder design for hardware reuse. At the same time, it provides the ability to begin high-level application coding before the final new I/O circuitry design is complete.

It can be problematic if driver software and the associated APIs do not execute properly or return device-specific errors without the I/O circuitry installed. To get around this problem, software developers often create simulation subroutines that temporarily replace the I/O circuit code within the application. This method makes it difficult to get a start on the application development and virtually impossible to system services have proven their mettle in thousands of deployed machine-builder applications. Parallel and multithread-safe embedded-middleware drivers are an integral part of RIO. The machine builder can call both multithread-safe and reentrant functions from multiple threads at the same time, and still operate correctly without blocking—an important feature for writing parallel code and optimizing performance. Drivers that lack reentrant execution can erode performance or, worse, cause a crash. Your code has to wait until the other threads are done using each function before it can access them. Reentrancy is an important consideration to eliminate any unnecessary dependencies in your code.

the PMSM's voltage is sinusoidally shaped.

Brushless DC motors are more costly than AC induction motors but provide better energy efficiency and performance when controlled using advanced algorithms. Moreover, they can scale up to serve very high-power and high-speed applications. While AC induction motors still dominate the market, brushless DC motor sales have quadrupled over the last five years to more than $1.2 billion, according to the ARC Advisory Group.

BLDC motors are a type of synchronous motor. This means the magnetic field the stator generates and the magnetic field off the rotor rotate at the same frequency. Usually BLDCs are equipped with three phases. The stator of a BLDC motor con-

sists of stacked steel laminations with windings placed in slots that are axially cut along the inner periphery. Most BLDC motors have three stator windings connected in a star fashion. The internal structure is like that of an induction motor containing pairs of permanent magnets on the rotor rather than windings.

As the name implies, the brushless DC motor is designed to operate without brushes. This means that the commutation the brushes provide must now be handled electronically. To rotate the BLDC motor, the stator windings are energized in a sequence. To calculate which winding to energize at a time, it is necessary to know the rotor position, typically measured by three Hall-effect sensors embedded into the stator. Based on the triple combination of these sensor signals, the control electronics can determine the exact sequence of commutation.

Because brushless motors use permanent magnets in their rotors rather than passive windings, they natively provide higher power than induction motors for their size and weight. The key to high-efficiency operation, however, lies in the FPGA-based controller.

FPGA-based algorithm control delivers better efficiency than microprocessors can achieve. A wide range of control-system algorithms are available, including trapezoidal, sinusoidal and field-oriented.

Trapezoidal, or six-step, control is the simplest but lowest-performance method. For each of the six commutation steps, the motor drive provides a current path between two windings while leaving the third motor phase disconnected. However, torque ripple causes vibration, noise, mechanical wear and greatly reduced servo performance.

## Motor Efficiency Leads to Greener Bottom Line

By Wil Florentino
Product Marketing Manager, ISM Vertical Market, Xilinx

On today's factory floor, the motor-driven equipment is estimated to be responsible for two-thirds of the total electricity-energy consumption. This is creating a challenge for equipment manufacturers to develop more energy-efficient systems. The effort can be manifested in many ways, such as selecting a more efficient motor, properly sizing a motor for its designed load or improving motor performance through better and faster feedback and control.

**Check the Efficiency Rating**

Simply replacing an existing motor with a more energy-efficient model justifies the slight premium of its initial cost. For a 500-horsepower motor that runs 8,000 hours a year, switching to a model with a 5 percent higher efficiency rating could save more than $12,000 and 170 kilowatt-hours of electricity annually. To offset the upfront costs, some utility companies and public agencies provide incentives to encourage customers to upgrade to NEMA Premium Motors, certified by the national trade organization for the electrical manufacturing industry.

**Size Matters; So Does Load**

In general, motor efficiencies also vary based on the amount of load on the motor, and can range from 85 percent to 97 percent at full load, with a peak on average at 70 percent to 80 percent load. Counterintuitively, slightly oversizing a motor (up to 25 percent) can actually increase efficiency. Also, a seemingly minor increase in a motor's full-load rotational speed of 40 RPM can result in a 3 to 6 percent increase in the load placed upon the motor, increasing energy consumption by 7 percent and offsetting the energy and dollar savings expected from a NEMA Premium Motor.

**Field-Oriented Control Squeezes More Savings**

The load to the motor is never constant; therefore, feedback from the motor will help to track incremental load changes. While the traditional, scalar control techniques for variable-speed operation of three-phase electric motors offer simple implementation, they limit the performance. Field-oriented control (FOC), also called vector control, is a better method, since it provides for faster feedback and tighter control of the torque by controlling the current with every PWM cycle. In this way, FOC ensures that current is inherently limited. An additional benefit is the ability to use a smaller motor without sacrificing torque or speed, while running at higher efficiencies, providing better dynamic response.

Apart from enabling the use of a smaller motor, FOC can lower a motor's energy consumption while providing better efficiency. For example, FOC implemented within an AC induction motor can improve motor efficiencies up to 25 percent beyond what's attainable in a non-field-oriented approach. The U.S. Department of Energy estimated that implementing energy-saving techniques, such as FOC, can shave operating margins by 1 to 5 percent, significant when compared with the typical plant operating margins of 16 percent.

**Science and Art**

There is a definite science and a little bit of art involved in designing-in the most efficient motor and drive system for your equipment. Choices can range from selecting the right size motor for the expected load to pushing the performance limits using a more-complex motor-control algorithm such as FOC.

The benefits of increasing the efficiency of motor-controlled systems start with lower manufacturing costs. Indeed, no matter whether you are moving fluids, using a drill or controlling a robot arm, motor selection and control techniques will affect your bottom line. Also, the environmental impact of reducing energy consumption provides for more efficient use of the limited natural resources available to us today.

**Field-oriented control, also known as vector control, improves upon sinusoidal control by providing high efficiency at faster motor speeds. It allows precise and responsive speed control when the load changes and also guarantees optimized efficiency.**

Sinusoidal control, also known as voltage-over-frequency commutation, addresses many of these issues. A sinusoidal controller drives the three motor windings with currents that vary smoothly. This eliminates torque ripple issues and offers smooth rotation. The fundamental weakness of sinusoidal commutation is that it attempts to control time-varying motor currents using a basic proportional-integral (PI) control algorithm, and doesn't account for interactions between the phases. As a result, performance suffers at high speeds.

Field-oriented control, also known as vector control, improves upon sinusoidal control by providing high efficiency at faster motor speeds. It delivers the highest torque per watt of power input compared with the other control techniques, and allows precise and responsive speed control when the load changes. FOC also guarantees optimized efficiency, even during transient operation, by perfectly maintaining the stator and rotor fluxes.

### A Deeper Look at FOC

One way to understand how FOC works is to form a mental image of the coordinate reference transformation process. If you picture an AC motor operation from the perspective of the stator, you see a sinusoidal input current applied to the stator. This time-variant signal generates a rotating magnetic flux. The speed of the rotor is a function of the rotating flux vector. From a stationary perspective, the stator currents and the rotating flux vector look like AC quantities.

Now, imagine being inside the motor and running alongside the spinning rotor at the same speed as the rotating flux vector that the stator currents generate. Observing the motor from this perspective during steady-state conditions, the stator currents look like constant values and the rotating flux vector is stationary. Ultimately, you want to control the stator currents to obtain the desired rotor currents. With coordinate reference transformation, you can control stator currents such as DC values using simple PI-control loops.

Under the hood, the FOC algorithm works by removing time and speed dependencies and enabling the direct and independent control of both magnetic flux and torque. It accomplishes this by mathematically transforming the electrical state of the motor into a two-coordinate time-invariant rotating frame using mathematical formulas known as the Clarke and Park transformations.

An efficient method to control the power electronics, called space-vector pulse-width modulation (PWM), maximizes the usage of the motor supply voltage and min-

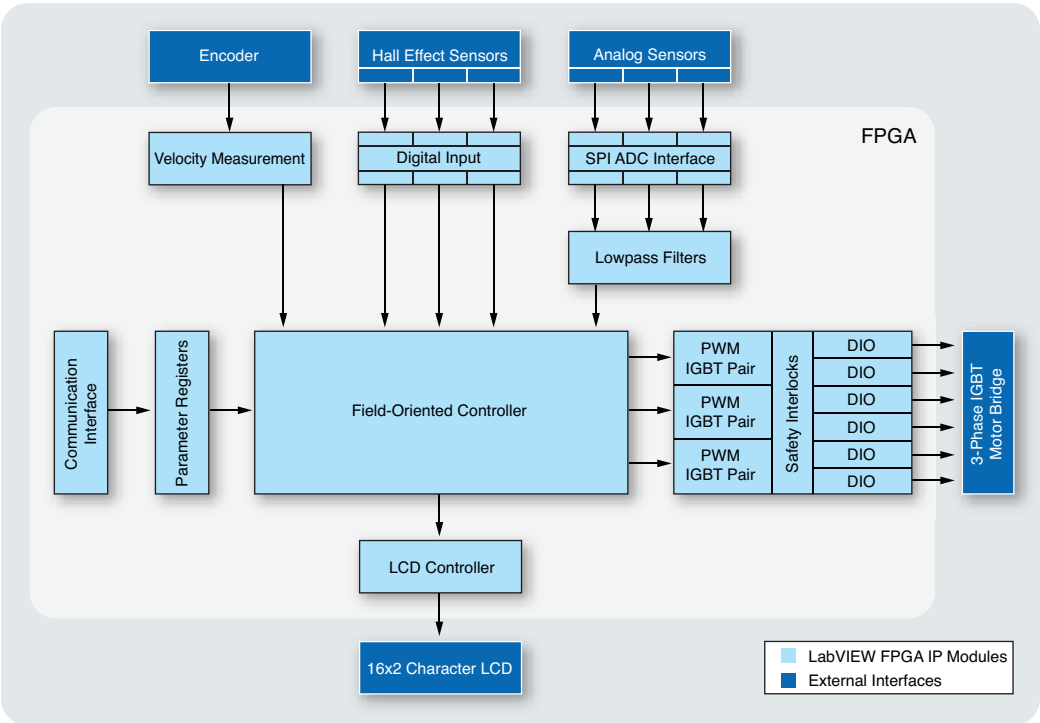



*Figure 4 – System diagram for FOC implementation using a Xilinx FPGA-based RIO hardware platform*

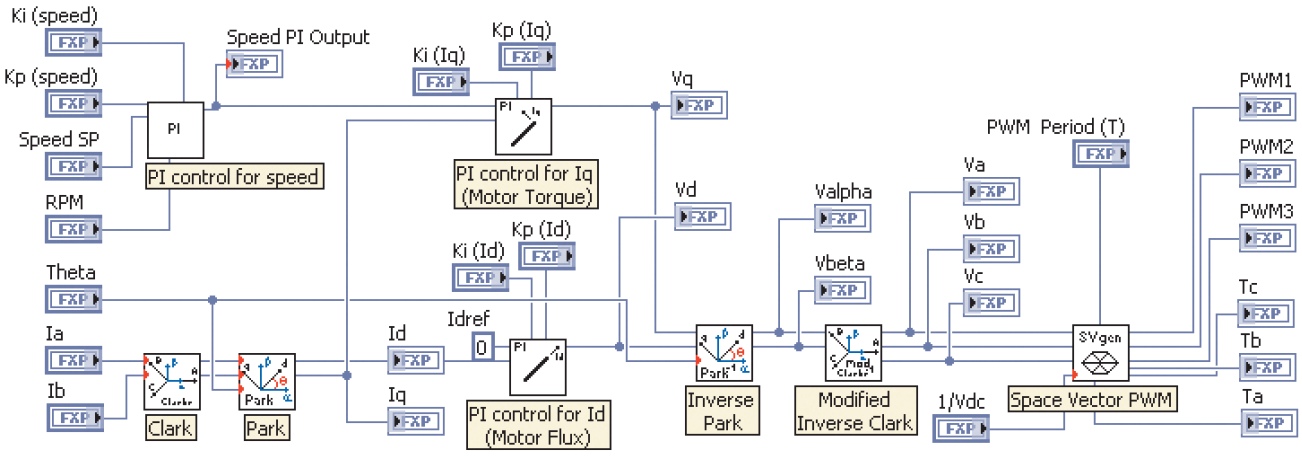



*Figure 5 – Field-oriented control algorithm for LabVIEW FPGA*

imizes harmonic losses. Harmonics can significantly erode motor efficiency by inducing energy-sucking eddy currents in the iron core of the motor.

Best of all, designers can utilize field-oriented control for both AC induction and brushless DC machines to improve efficiency and performance. They can also apply FOC to existing motors by upgrading the control system. In fact, they can employ vector-control techniques like FOC with AC induction motors to enable servo-motor-like performance.

## FPGAs Tackle FOC Challenge

It takes powerful computation devices to implement FOC, and this requirement makes FPGAs a natural fit for motor control. An FOC system must continuously recompute the vector-control algorithm at a rate of 10 to 100 kHz. In parallel to the control algorithm, additional intellectual-property (IP) blocks such as the high-speed PWM outputs need to execute without affecting the timing of the control algorithm. With their inherent parallel execution and hardware reliability, FPGAs are able to perform control algorithms with loop rates up to hundreds of kilohertz, with room left over to handle communication and provide the data for user-interface applications on the host microprocessor. Moreover, the reconfigurability of FPGAs allows the customer to adjust the control algorithm whenever necessary.

Figure 4 illustrates a system for FOC implementation using a Xilinx FPGA on the National Instruments RIO platform. Besides the actual control algorithm, the FPGA executes IP blocks to read the three Hall-effect sensors, an encoder and three additional analog sensors as it generates PWM signals that drive external electronics to power the motor. For communication to a host processor and a simple user interface, IP blocks execute in parallel.

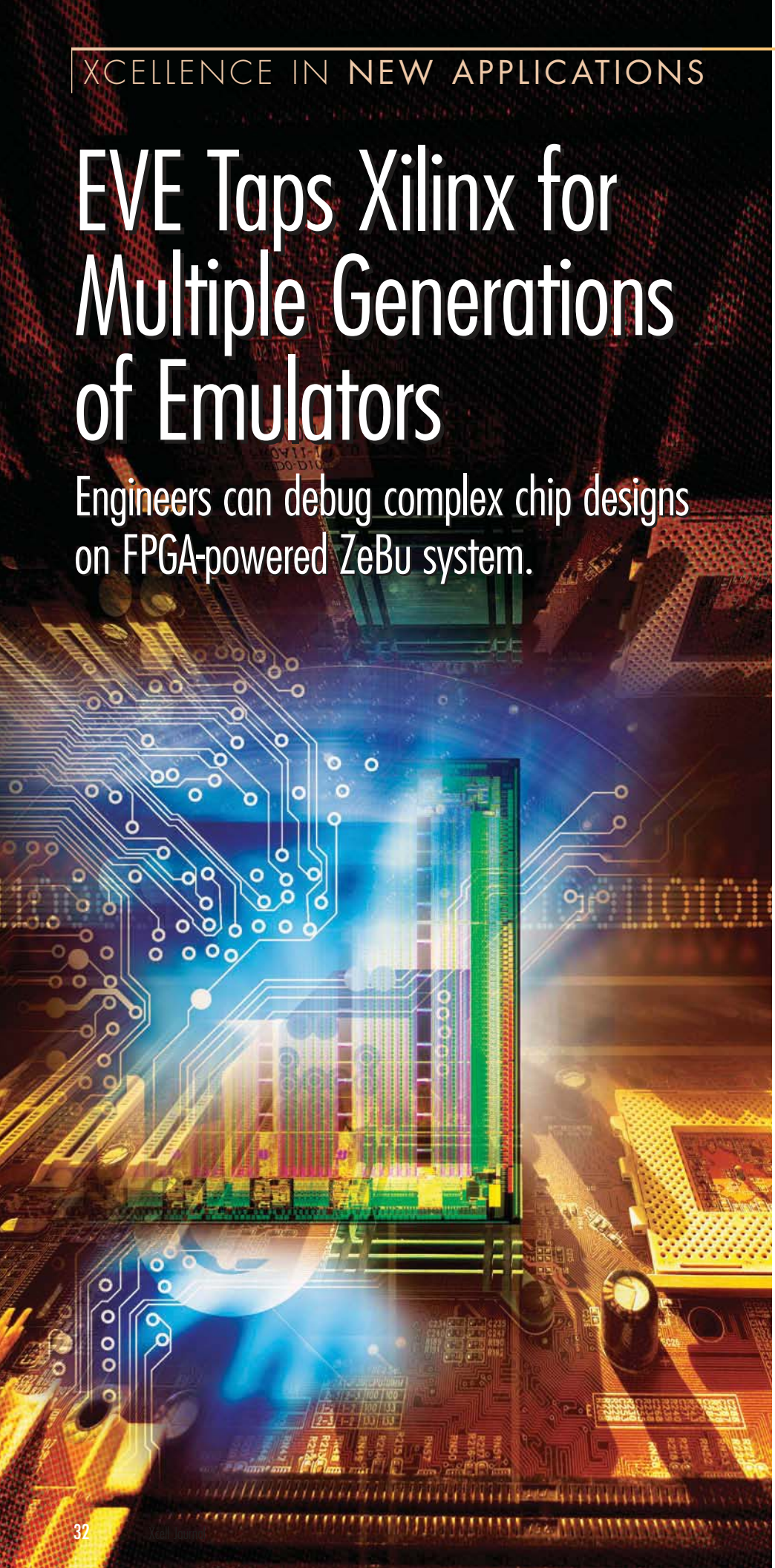
Figure 5 shows the LabVIEW FPGA implementation of the FPGA-based FOC algorithm. The Clarke transformation converts the three-axis coordinates shifted by 120° (Ia, Ib, Ic) into orthogonal two-axis ones (Ia, Ib). In a second step, the Park transformation converts the fixed (Ia, Ib) coordinates into decoupled two-axis rotating coordinates (Id and Iq), which a simple PI controller can then control. The FOC system uses inverse Park and Clarke transformation to bring them back into the fixed AC three-phase frame of the stator windings. NI provides the complete source IP at our IPNet Web site at *ni.com/ipnet*.

One of our customers, electronic-system designer and manufacturer BAE Systems Avionics, used our RIO platform with Xilinx FPGAs to squeeze 15 percent extra performance from its existing motors while saving weight in avionics products by reducing motor mass. Thanks to the efficiency and tight power control of FOC, the BAE Servo Systems Technology Group in Edinburgh, Scotland, now specifies smaller motors than previously possible.

Ultimately, machine-builder design requirements insist on improving motor operating efficiency to meet regulatory restrictions and to ensure a rapid return on investment. When evaluating control-system upgrades, machine designers often underestimate energy costs, calculating that they are typically orders of magnitude higher than hardware costs over the life cycle of the motor. We at NI focus our efforts on delivering flexible embedded controllers with high computing performance using commercial off-the-shelf hardware solutions based on FPGA technologies from Xilinx. With this combination, we can meet the most extreme customer challenge, specifically the performance requirements of FOC.

# EVE Taps Xilinx for Multiple Generations of Emulators

## Engineers can debug complex chip designs on FPGA-powered ZeBu system.



by Ludovic Larzul
Vice President of Engineering
EVE
*ludovic_larzul@eve-team.com*

Creating a hardware emulation system is no easy task. At a minimum, each generation of emulation system has to accommodate a growing number of logic gates, memory and DSP blocks to allow ASIC and ASSP system-on-chip (SoC) designers to debug their extremely complex devices before sending them off to the foundry for production. Emulation systems must also be easy to program, reliable and, what's more, affordable. Here at EVE, we've developed several generations of emulation systems—leveraging the power of Xilinx® FPGAs—to emerge as a leader in the market.

Today's SoCs are exceedingly complex pieces of silicon. They contain one or more processors that will execute software. The software code they run is every bit as important a part of the final system as the silicon itself. The software and the silicon have to act as a seamless solution; if there's a problem, it might be the software, or it might be the silicon.

Designers can only do so much software testing on a development host. No reasonable host development system can reflect the true parallelism of the target SoC. You can really only test out such issues as synchronization, data integrity and resource contention in situ, and that's far too late to identify problems. Simulation isn't a viable solution; it's simply too slow to allow the execution of any realistic code.

As a result, engineers have been using emulation systems for well over two decades to verify the most advanced ICs the semiconductor industry can build. Most of these earlier-generation emulation systems were powered by custom ICs that the emulator vendors designed themselves. They would then pass the cost of the custom IC development on to their customers, making the power of emulation more cost-prohibitive for companies struggling with ever-tighter IC development budgets.

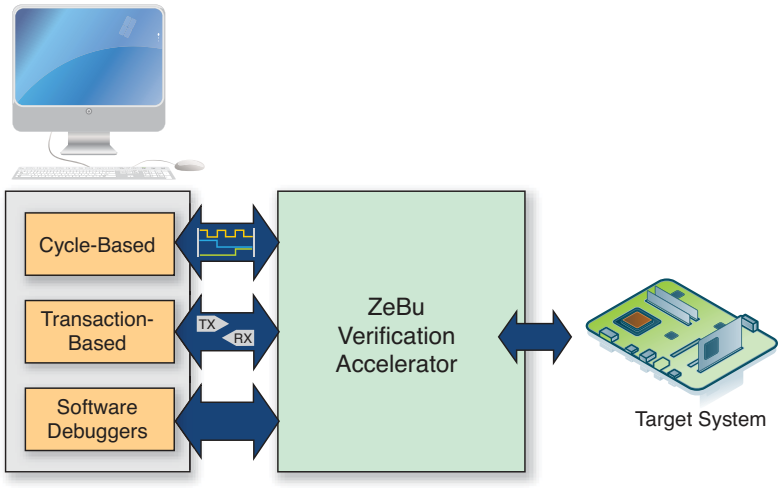



*Figure 1 – EVE bases its ZeBu (for "zero bug") emulation system around Xilinx FPGAs.*

In 2001, EVE broke with tradition by basing our innovative emulation system on Xilinx FPGAs. The goal was to provide the lowest hardware-assisted verification cost of ownership in the industry, as achieved through a combination of high execution speed, high capacity (which today means up to a billion gates), quick design revision, flexible and powerful debugging capabilities, lowest cost per gate and most cycles per dollar. In addition, we wanted to make the system easy to use for ASIC designers who might not be familiar with FPGA design.

The result was the ZeBu (for "zero bug") emulation system (Figure 1). We've now developed six generations of emulators, the most recent of which is ZeBu Server (see sidebar), and we're still going strong.

## Separation of Powers

Our approach is to split the device-under-test (DUT) from an interface to the test environment that we call Reconfigurable TestBench (RTB). The RTB allows for test configuration and control. The DUT will change with each rev of the design, but the RTB never changes unless the test environment does. Having a single mass of FPGAs containing a mix of the RTB and DUT designs would have been messy and required unnecessary recompilation of the RTB design, so we separated them out.

As a result, we have one set of FPGAs for the DUT and another set for the RTB (Figure 2). The number of DUT FPGAs varies by system size; bigger and smaller systems are available for bigger and smaller designs.

The RTB FPGAs provide communication and control. We can optimize them much more aggressively for performance and capacity, since the design will remain constant. While no one reconfigures the contents of this set of FPGAs, the design allows a rich set of configurations of the test setup under user control.

For the DUT FPGAs, the overwhelming priority in terms of required features was density. We needed to be able to put really big designs in here in order to provide value. Key among the necessary hardware features were multipliers, which later gave way to DSP blocks. In addition, big designs needed

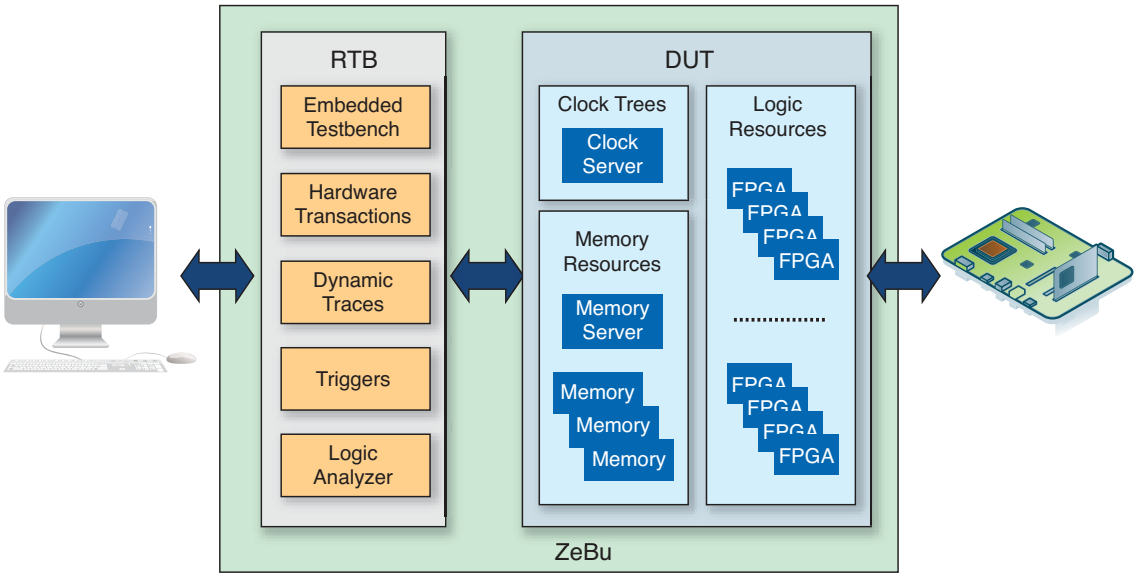



*Figure 2 – ZeBu uses one set of FPGAs for the device under test and another for the Reconfigurable TestBench (RTB).*

big memory, so the largest possible memory blocks were a requirement.

Close behind density on our "must have" list were bandwidth and latency. The data transmission between the FPGAs had to be fast enough to keep from becoming a serious bottleneck. We didn't plan to use clock-data-recovery (CDR) circuits—in fact, they weren't even available when we started out—but we did use high-speed LVDS I/Os, layering our own proprietary low-latency protocol over them.

To provide robust debugging capabilities, we needed readback of internal states. We wanted a way of interrogating what was going on inside so that designers could understand the inner workings of their technology—especially when it didn't seem to be working. We might implement readback using JTAG or some other means, but it had to be there.

Finally, to keep the challenge of implementing a single design across multiple FPGAs tractable, we wanted to use only a single technology on the board and across different members of the emulator family. So the FPGA family we chose had to have a broad range of densities and pin counts.

Our considerations for the RTB FPGAs were different. Again, one of the primary requirements was keeping to one FPGA technology per system. This would ensure that all of the FPGAs were in sync with respect to version and availability. Of course, we still needed to be able to meet our performance requirements, no easy task given that the RTB FPGAs run at twice the speed of the DUT FPGAs.

Given the sum of requirements, the Virtex®-II family was our clear choice for the first ZeBu generation. At the time, it led in gate and memory density, provided greater bandwidth with less latency and excelled in its readback capability. Our first system used two Virtex-II 8000s for the DUT and one Virtex-II 6000 for the RTB.

Over time we've continued to evaluate the Virtex family against others as the technology has advanced, but to date we have never found another choice compelling enough to move us away from the Virtex. As such, we have progressed with Virtex-4 (up to 64 Virtex-4LX220s for the DUT) and Virtex-5 (as many as 400 Virtex-5LX330s), and are eyeing the new Virtex-6 family with eager interest.

### Going With a Flow

We design the RTB FPGAs with a standard FPGA flow, using XST for synthesis and the standard Xilinx ISE® tools for place and route. The effort we make here is typical of what anyone designing with FPGAs might do to create a complex design. The differentiation isn't in the flow; it's in the hard work and our system knowledge.

When it comes to the DUT flow, however, our challenge is far greater than the one the typical FPGA designer faces. By

## Evolution of Hardware Emulation Culminates in ZeBu-Server

Hardware emulation has become a valued component of the hardware/software co-design flow, enabling hardware engineers and software developers to share system and design representations and work together to debug hardware/software interactions.

Use of this tool as a popular EDA solution has evolved over the past 20 years, from the early days of standard FPGA-based emulators to custom ASIC-based models, back again to emulators based on standard FPGAs.

CPU and graphics chip engineers were the first to use emulation, because the sheer complexities of their designs crippled traditional event-based hardware description language (HDL) simulators. The tool quickly gained acceptance by the wireless community due to the extensive use of embedded software in its hardware designs. Today, consumer electronics companies widely use hardware emulation for the design of digital TVs, set-top boxes, digital still cameras and camcorders, multifunction printers and other products.

The early version of hardware emulation delivered execution speeds four to five orders of magnitude faster than simulation, making it ideal for accelerating the time required to develop and validate the hardware of ASIC or system-on-chip (SoC) designs. However, it did not meet the minimum speed of execution required for efficient testing of embedded software—namely, 1 MHz. Moreover, cost of ownership hampered emulation's popularity and restricted its adoption to large corporations in limited numbers.

The newer FPGA-based emulators are more reasonably priced and cost-effective, and have shortened the overall verification cycle of complex chip and electronic-systems designs. These emulators have a smaller footprint than prior emulation tools, are fast, efficient and easy to use.

Setup is straightforward and newer emulators consist of fewer FPGAs than older machines. The latest generation of emulation systems can execute billions of verification cycles, as required in embedded designs, in a short period of time. They provide a full view of the design, necessary to debug the hardware. These machines also support transaction-level verification, which is needed for hardware debugging at a high level of abstraction, via monitors, checkers and assertions.

EVE is an FPGA-based emulation trendsetter. On July 14, it launched the latest incarnation of the ZeBu (for "zero bugs") product line, called ZeBu-Server, a sixth-generation version of its emulator based on the Xilinx Virtex LX330.

Providing design capacity of up to 1 billion ASIC-equivalent gates, ZeBu-Server can be used across the entire development cycle for verifying and debugging a new ASIC or SoC design. Designers can use it to test the integration between hardware and software, and to validate embedded software before silicon availability.

This new emulator improves on previous-generation offerings in terms of capacity, speed, setup time, integration and debugging capabilities. And last but not the least, it is also cost-effective.

*— Ludovic Larzul*

definition, we deliver an unfinished system, in the same way that Xilinx delivers an unfinished chip. Our customers have to fill in the DUT design details. So we must provide them with a way to implement their designs—while assuming that they may not know or care about how to do FPGA design. To help facilitate this, we created the ZeBu Compilation User Interface (zCUI).

A key concern is the fact that the DUT design, by definition, will change numerous times throughout the design cycle. Minimizing the design turn time has always been a high priority. Because the first systems were relatively small, we didn't focus on the compiler performance; we simply worked through the standard Xilinx flow. But as we moved up in density, compile time—and, in particular, synthesis time—became more critical. A design that will occupy 400 of Xilinx's largest FPGAs is not trivial to synthesize.

We were concerned that standard synthesis tools were spending too much time doing too good a job. Our priority was not optimal synthesis—optimal meaning highest performance, lowest gate count. Those are good things, but for us, compile time was a higher priority, and we could live with 20 percent worse performance or density—especially since we're executing at less that 30 MHz.

For this reason we developed our own zFAST synthesis tool, which could take SystemVerilog, VHDL or Verilog designs and synthesize them as much as 10 times more quickly than standard synthesis products. We now make zFAST available alongside the more traditional synthesis tools so that our customers can invoke it for large designs as needed. Once the design is synthesized, our own partitioning tool partitions it at the gate level.

It should be noted that some of our users are accomplished FPGA designers, and we willingly hand over greater control if they want it. The customer is free to use the standard FPGA flow to try to get more performance out of the DUT, constraining and using any tricks they know as necessary. So the system makes it easy for a non-FPGA designer, but does not restrain an expert.

The continuing performance improvements of the ISE tools from Xilinx have also helped our overall compile times. EVE was one of the first users of ISE 11, allowing us to keep our FPGA compile times typically under two hours per FPGA. Parallel compilation means that we can turn large designs with multiple FPGAs in a reasonable amount of time. Our overall flow is summarized in Figure 3.

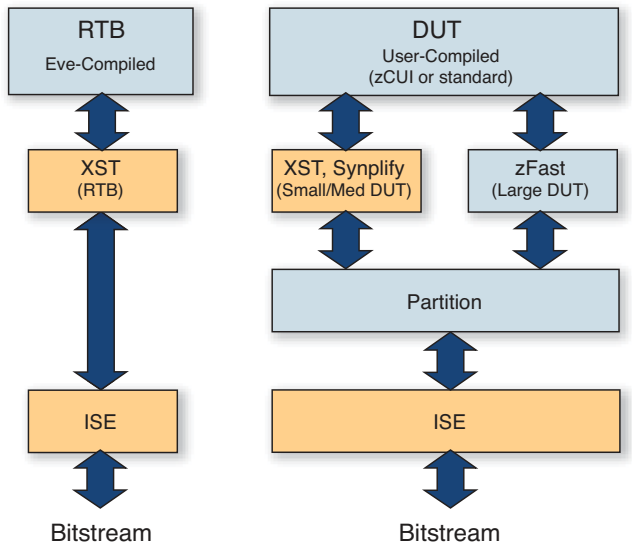



*Figure 3 – The ZeBu design flow makes use of Xilinx's ISE tools.*

## Trust, but Verify

Verification is a bit tough for us, specifically because we don't ship a finished design. We have to be confident that, once our customer compiles a design onto the ZeBu platform, everything will work as promised.

While the verification of our first set of RTB FPGAs was a challenge, since then we've been able to use our own existing ZeBu systems to verify the RTB for the next-generation ZeBu system. We run the new RTB for hours and days, executing billions of cycles, to confirm that it's solid.

For the DUT FPGAs, in conjunction with our own tools we have a large vault of designs accumulated over years of experience. Each night, we run thousands of designs and test the results using application-specific test bench suites associated with each individual design, as well as through cycle-by-cycle simulation comparisons that include internal state as well as inputs and outputs.

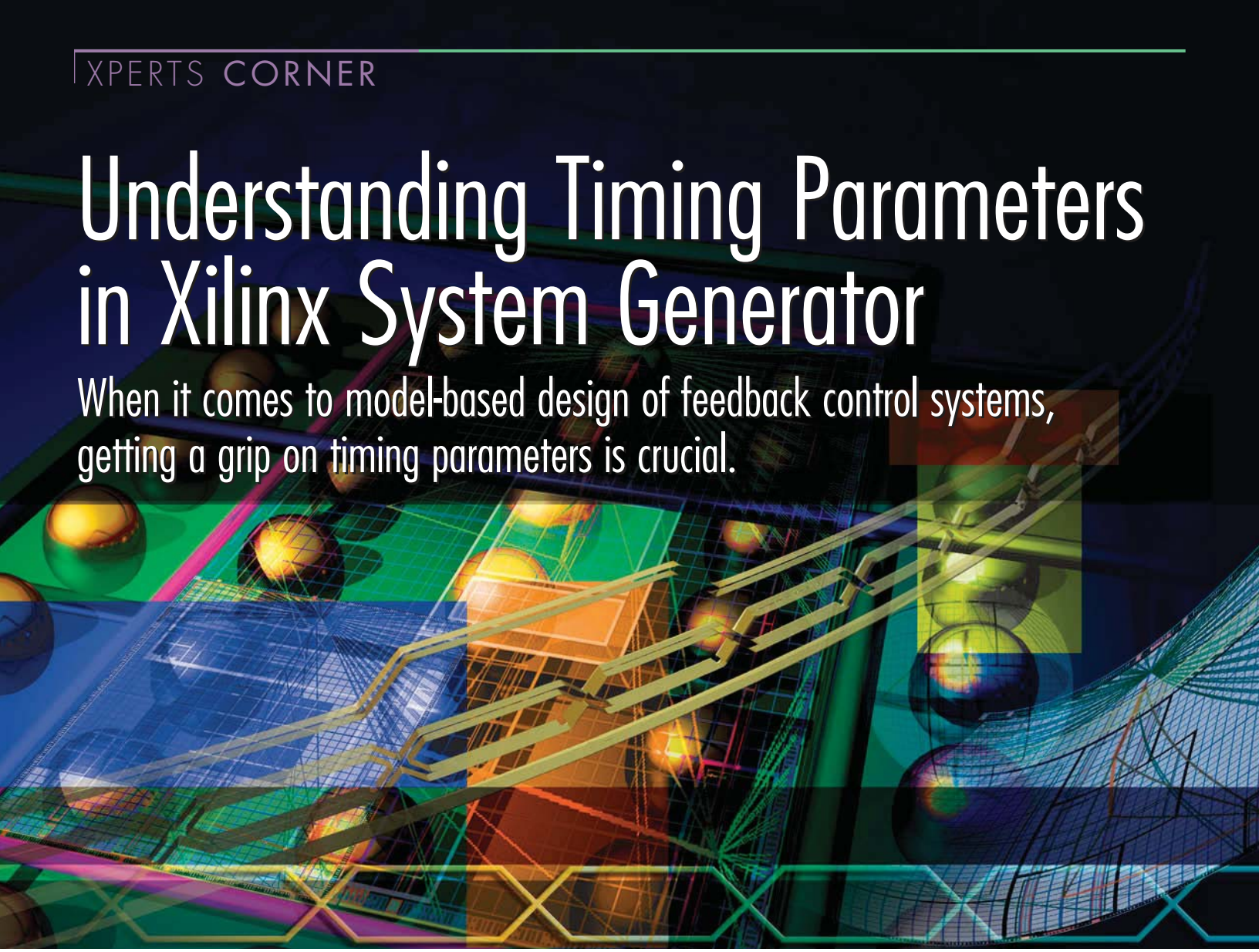
We also check to make sure that our results remain deterministic as our tools and those from Xilinx evolve. Whenever there's a revision change, we run through the suite of tests to ensure that the results we get with the new version are the same as those we got on the older version.

Looking back, then, it is clear that EVE's technology has been intricately bound up with Xilinx's technology. Starting with the original choice of Virtex-II, we have dovetailed the growing capacity of the Virtex chips and the evolution of the Xilinx tools with our own system design and tools to provide a solution that continues to gain traction in the emulation market. Given the Xilinx and EVE road maps, that symbiosis is likely to continue unabated.

*Ludovic Larzul, a founder of EVE and its vice president of engineering, has 13 years of experience in CAD development. He has worked on numerous projects, including the integration between hardware emulators and software simulators, efficient interfaces based on transactors, communication among multiple emulation systems and system-level place and route. Larzul holds a Diplome d'Ingenieur de l'Institut de Recherche et d'Enseignement Superieur aux Techniques de l'Electronique (Ecole Polytechnique de Nantes).*

# Understanding Timing Parameters in Xilinx System Generator

## When it comes to model-based design of feedback control systems, getting a grip on timing parameters is crucial.



by Juergen Wassner
Lecturer
Lucerne University of Applied Sciences and Arts,
School of Engineering and Architecture
*juergen.wassner@hslu.ch*

Christoph Eck
Lecturer
Lucerne University of Applied Sciences and Arts,
School of Engineering and Architecture
*christoph.eck@hslu.ch*

Model-based design (MBD) has recently attracted much attention for its promise of closing the gap between abstract mathematical modeling and physical realization of real-time systems. By using the same source for algorithm analysis, architecture exploration, behavioral simulation and hardware/software design, MBD promises to shorten the system design cycle.

Several tools for model-based design have emerged over the past years, many of which center on the MATLAB® and Simulink® environments from The MathWorks. An elegant way to leverage the precious stimuli-generation and data-analysis features of MATLAB/Simulink for MBD is by employing the Xilinx System Generator for DSP tool. Despite its name, this tool can be useful for disciplines other than classical digital signal processing. Control applications are of particular interest, since MATLAB/Simulink is the standard design and simulation environment for the control-engineering community.

Xilinx System Generator for DSP enables control engineers to design their system in the familiar Simulink environment and then implement it in an FPGA, without knowledge of a hardware description language (HDL). To do so, you must relate the parameter values in the mathematical model of the controlled system (often called the plant)—for example, a continuous- or discrete-time transfer function or state-space description—to the sample rate of the digital controller and the FPGA system clock frequency.

### Digital Controllers in FPGA

As with any signal-processing algorithm, digital controllers can be implemented in either software or hardware. A combination of the two—that is, a hardware-accelerated software implementation—is rarely necessary, due to the moderate sample-rate requirements and algorithm complexity of most control applications. When choosing FPGAs as implementation technology for digital controllers, designers can utilize features not available with MCU-based software solutions without compromising on

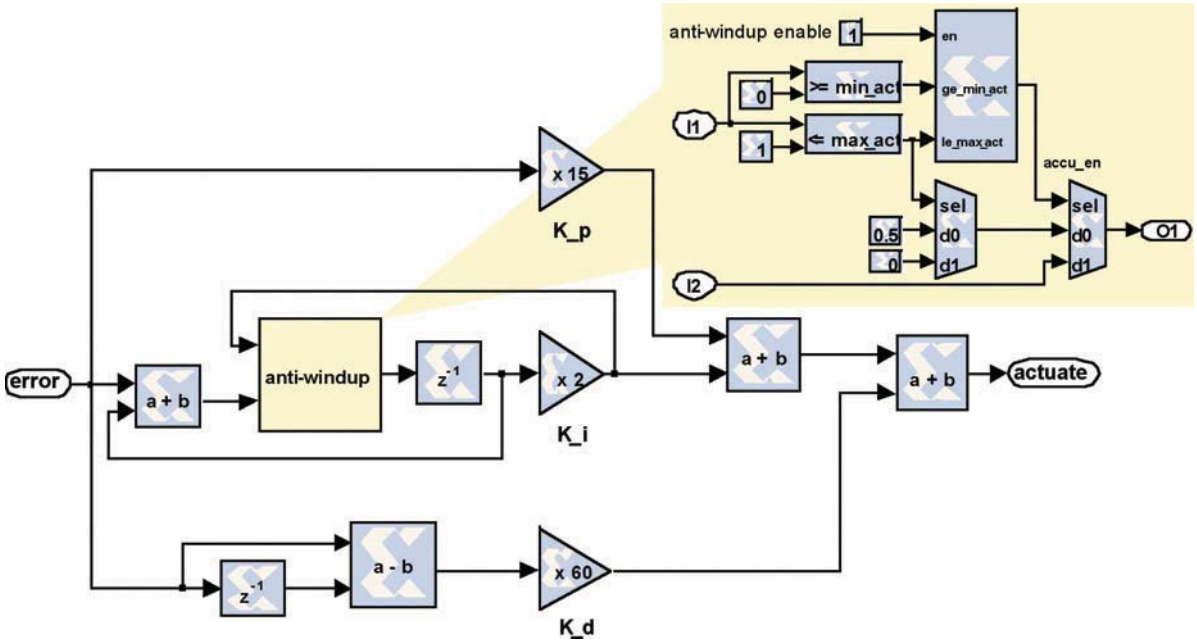



*Figure 1 – PID controller with anti-windup function built from System Generator blocks*

field-upgradability. The non-interrupt-driven, parallel data processing in FPGAs guarantees absolute timing determinism even when multiple processes with different sample rates need to be controlled. This, together with configurable word widths, makes FPGAs an attractive choice for the implementation of digital controllers.

Traditionally, for FPGA implementation, the controller designer would use a low-level HDL after first validating the control strategy and parameters with high-level simulations of controller and plant models, using for instance Simulink. An HDL testbench will verify correspondence between the HDL controller design and Simulink simulation. And in order to verify the controller design in a closed-loop system, this testbench must contain a model of the plant. Accomplishing all this is no easy task for designers, including most control engineers, who lack a sound background in HDL and FPGA technology. In this situation, a high-level modeling and design environment such as Xilinx System Generator is the ideal solution.

## PID Controller in System Generator

Since many controllers are still based on the classical proportional–integral–derivative, or PID, structure, we have chosen a PID controller for demonstrating our points. Nevertheless, the approach we outline can handle other commonly used control components such as lead-lag compensators, state-space observers or adaptive controllers equally well. Figure 1 shows the PID controller we designed with blocks from the Xilinx block set.

Instead of using the Xilinx Accumulator block, we accomplished integration with the elementary building blocks Adder and Register. Doing so enables us to insert anti-windup logic, also shown in Figure 1, to freeze the contents of the accumulator register whenever the integral part of the controller output reaches the saturation limits that the actuator imposes. The anti-windup logic makes the PID controller a nonlinear system and has a positive effect on the overall system dynamics.

The block parameter menu shown in Figure 2 can configure the control parameters and word widths of various signals. Also, designers can enable or disable the anti-windup function here. This menu provides for convenient experimentation without the need to modify low-level HDL code.

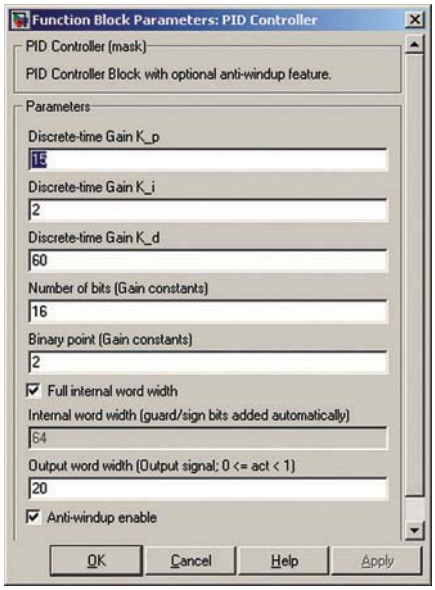



*Figure 2 – Customized parameter menu for the PID controller*

Figure 3 shows the overall system model, which contains, besides the controller, the plant and the simulation testbench built from standard Simulink blocks. Here designers can model the plant by either a continuous- or discrete-time transfer function, whereas in an HDL testbench only discrete-time functions would be appropriate. Note that in the System

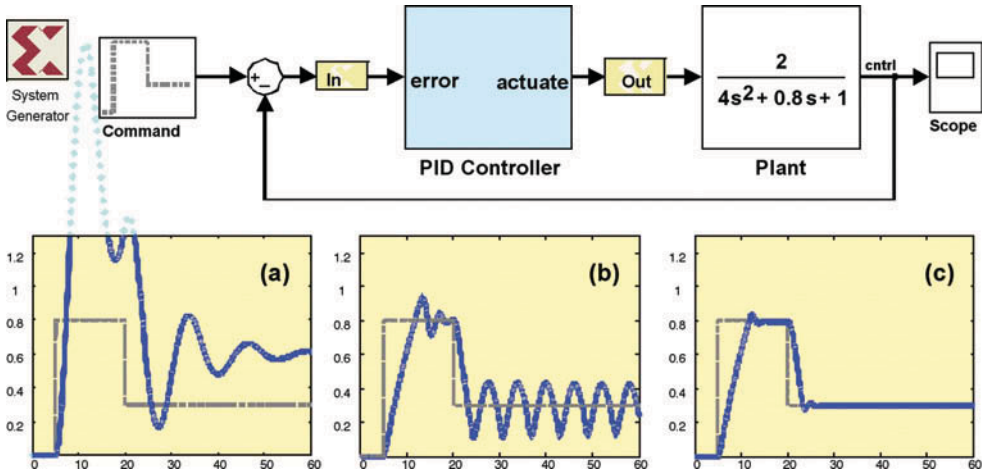



*Figure 3 – Overall system model (top) and plant output in response to command input without control (a), and with PID control and anti-windup disabled (b) and enabled (c).*

Generator approach, you can perform anything from system modeling through simulation and verification, up to implementation, using one and the same high-level model.

### Understanding Timing Parameters

Now let's look at the various timing measures you will come across during this process. In what follows, we deliberately make use of the same terminology as the Xilinx System Generator documentation, even if this terminology does not always seem to be intuitive.

There are several types of timing parameters. Those that refer to absolute units such as megahertz or nanoseconds are denoted by upper-case letters. All other parameters are denoted by lower-case letters. Furthermore, the timing measures can be divided into control and analysis parameters. All these parameters are summarized in Table 1.

### Control Parameters

The first of the control parameters is the simulation time unit $T_{Sim}$. You must not enter this unit explicitly anywhere in your design. It represents your implicit assumption on the fundamental unit of time in Simulink simulation. As such, it affects simulation solely. In the Simulink as well as System Generator context, the simulation time unit is often assumed to be one second. For instance, the display of the System Generator WaveScope block uses this convention. But as we will see below, $T_{Sim}$ can be any other time unit that suits your needs.

You must set the next parameter, the FPGA clock period $T_{CLK}$, in the System Generator token in units of nanoseconds. It represents the period of the main system clock input to the FPGA from which all other clock and clock enables are derived. Hence, its setting affects only hardware implementation. For instance, for the popular Spartan®-3E Starter Kit from Xilinx, the FPGA clock period would be 20 ns (50 MHz).

The Simulink system period $p_{sys}$, meanwhile, represents the global link between

| | Name | Symbol | Unit | Effect / Scope | Location |
|---|---|---|---|---|---|
| Control | Simulation time unit | $T_{Sim}$ | s, ms, _s, ns | Simulation only, global | Implicitly assumed |
| | FPGA clock period | $T_{CLK}$ | ns | HW only, global | Enter in SysGen token |
| | Simulink system period | $p_{sys}$ | $[T_{CLK}/T_{sim}]$ | Simulation and HW, global | Enter in SysGen token |
| | Sample period | $p_{sam}$ | $[T_{sim}]$ | Simulation and HW, local | Enter in Gateway-In block |
| Analysis | Sample time | $t_{sam}$ | $[T_{CLK}]$ | None, for analysis only | Displayed by ST block |
| | Sample frequency | $F_{sam}$ | MHz | None, for analysis only | Optionally displayed for each block port |

*Table 1 – Timing parameters for System Generator designs*

**In the System Generator approach, you can perform system modeling through simulation and verification, up to implementation, using one and the same high-level model.**

Simulink simulation and hardware implementation. Designers must set this parameter, which influences both Simulink simulation and hardware implementation, in the System Generator token. During simulation, this value determines how often the System Generator blocks of your model are called, but not necessarily updated, relative to the simulation time unit. For hardware implementation, it specifies the amount of overclocking with respect to the sample rate of the controller. Unlike the System Generator documentation, we define the Simulink system period as a unitless quantity—that is, the ratio of the FPGA clock period and the assumed simulation time unit:

$$p_{sys} = \frac{T_{CLK}}{T_{Sim}}$$

This makes it possible to assume arbitrary simulation time units, as mentioned above.

The sample period psam of a particular signal within the System Generator portion of a design is either set explicitly (for example, in the Gateway-In block) or is derived from rate-changing blocks such as Up Sample or Down Sample. When setting it explicitly, you would enter it as a numerical value in units of the assumed simulation time unit. Its setting has an impact on both Simulink simulation and hardware implementation. During simulation, this value determines how many calls to a block must occur before this block actually can change states. Similarly, in the implementation, this value represents the number of clock cycles after which the block logic will be enabled. And since all clock enable signals in a System Generator design are derived from the main FPGA clock input, each enable period must be an integer multiple of the FPGA clock period.

### Analysis Parameters

In the second category of timing parameters, namely, analysis parameters, the first one to consider is the sample time (ST) block. It uses no hardware resources in the implementation of the system, but solely serves analysis purposes in the Simulink model. The value $t_{sam}$ that the ST block displays is the clock enable period in units of FPGA clock periods used for the associated signal in the hardware implementation.

When designers select the option for the next analysis parameter, sample frequency, in the Icon Display property box within the System Generator token, each Xilinx block in the model displays the sample frequency $F_{sam}$ in megahertz, used in the implementation of this block. This sample frequency is related to the other timing parameters as follows:

$$1/F_{sam} = T_{CLKenb} = p_{sam} \cdot T_{Sim} = \frac{p_{sam}}{p_{sys}} \cdot T_{CLK} = t_{sam} \cdot T_{CLK}$$

where $T_{CLKenb}$ is the period of the associated clock enable in the implementation. Figure 4 exemplifies these relations.

From the second equation above it becomes clear that each sample period psam must be an integer multiple of the Simulink system period $p_{sys}$, since only those clock enable signals can be derived from the FPGA system clock. The third equation shows that the value the ST blocks display is the clock enable period in units of FPGA clock periods.

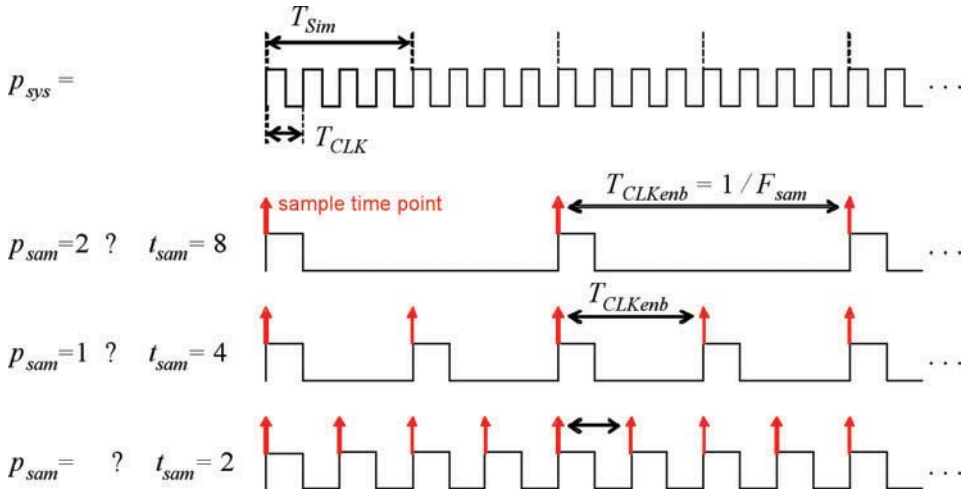



*Figure 4 – Relation of the six timing parameters exemplified for $p_{sys}$ = 1/4 and three different sample rates*

### Step-by-Step Guide to Choosing Timing Parameters

The example control system developed above offers insights into how to select timing parameters. We have broken the process down into five steps.

1. **Identify the plant.**

   Model the plant by an appropriate transfer function. In our example, we modeled the plant as a PT2 element with a gain factor K = 2, a time constant T = 20 ms and an attenuation factor d = 0.2. Hence, the plant is an oscillatory element, as can be seen in Figure 3(a).

2. **Choose the simulation time unit.**

   At this point you can choose the fundamental simulation time unit $T_{sim}$, such that the plant transfer function has convenient numerical parameters. In our working example we chose $T_{sim}$ = 10 ms. With the above parameters this yields the plant transfer function

$$H_P(s) = \frac{K}{T^2 \cdot s^2 + 2 \cdot d \cdot T \cdot s + 1} = \frac{2}{4 \cdot s^2 + 0.8 \cdot s + 1}$$

3. **Set the Simulink system period.**

   Given the simulation time unit, we are next going to set the Simulink system period $p_{sys}$ depending on the FPGA clock period $T_{CLK}$ of the hardware platform at hand. In the case of the Spartan-3E Starter Kit with 50-MHz system clock, we have $T_{CLK}$ = 20 ns and

$$p_{sys} = \frac{T_{CLK}}{T_{Sim}} = \frac{20\,\text{ns}}{10\,\text{ms}} = 2 \cdot 10^{-5}$$

4. **Determine the sample frequency.**

   A rule of thumb is that the sample rate of the digital controller must be at least 20 times larger than the cutoff frequency of the plant. Our example plant has a cutoff frequency of about 30 Hz and we thus opted for a sample frequency of $F_{sam}$ = 1 kHz.

5. **Set the sample period.**

   Finally, we set the sample period parameter $p_{sam}$ in the Gateway-In block in front of the controller. In the working example we have

$$p_{sam} = \frac{1}{F_{sam} \cdot T_{Sim}} = \frac{1}{1\,\text{kHz} \cdot 10\,\text{ms}} = 0.1$$

With these settings, we can simulate the model, tune the controller parameters and synthesize the controller logic. However, sometimes the FPGA clock period $T_{CLK}$ is much smaller than the fundamental time unit $T_{sim}$, for instance because your controller is part of a larger design that requires a much higher clock frequency than the controller itself. The simulation run-time can then become unbearably long due to the high number of void clock cycles to be simulated before the controller block actually processes the next data sample. In this situation you can use different settings for $p_{sys}$ in simulation and implementation without losing the consistency of your model. This is possible because the value of $p_{sys}$ affects only the System Generator part of your model.

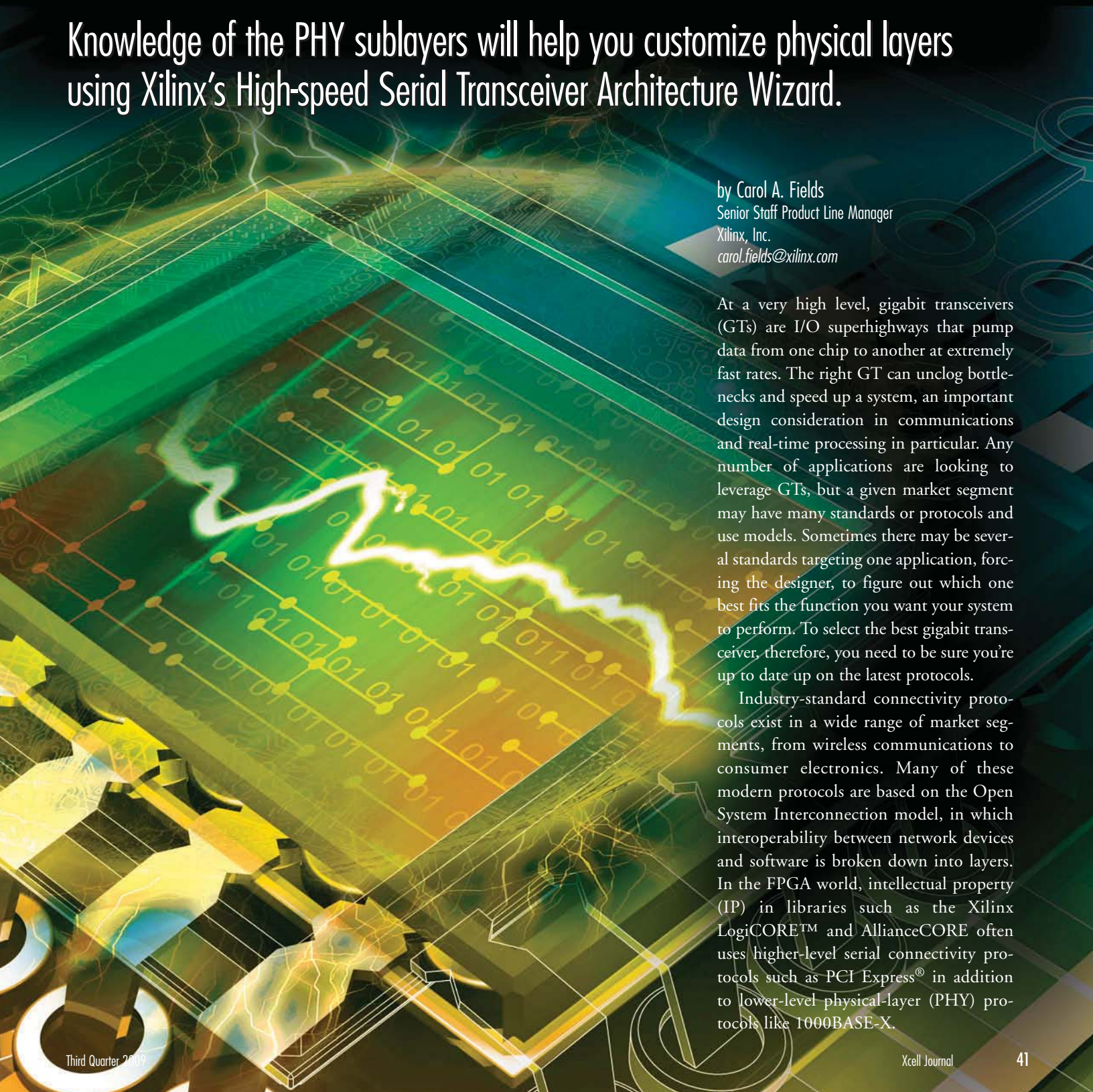
More specifically, you can set $p_{sys}$ = $p_{sam}$ during simulation of your control system. This ensures that the System Generator blocks are called only when necessary—namely, when the blocks actually can change states. Before you generate the FPGA implementation, you just need to switch back to the original value of $p_{sys}$.

Model-based design of closed-loop control systems requires that absolute timing measures of the plant transfer function be consistently related to the timing parameters of your design environment. We have provided a systematic approach to this problem using the Xilinx System Generator for DSP tool.

As a next step, we are going to investigate the hardware co-simulation features of Xilinx System Generator in order to perform real-time analysis of closed-loop systems with FPGA-based controllers.

# Right FPGA Gigabit Transceiver Makes All the Difference

## Knowledge of the PHY sublayers will help you customize physical layers using Xilinx's High-speed Serial Transceiver Architecture Wizard.



by Carol A. Fields
Senior Staff Product Line Manager
Xilinx, Inc.
*carol.fields@xilinx.com*

At a very high level, gigabit transceivers (GTs) are I/O superhighways that pump data from one chip to another at extremely fast rates. The right GT can unclog bottlenecks and speed up a system, an important design consideration in communications and real-time processing in particular. Any number of applications are looking to leverage GTs, but a given market segment may have many standards or protocols and use models. Sometimes there may be several standards targeting one application, forcing the designer, to figure out which one best fits the function you want your system to perform. To select the best gigabit transceiver, therefore, you need to be sure you're up to date up on the latest protocols.

Industry-standard connectivity protocols exist in a wide range of market segments, from wireless communications to consumer electronics. Many of these modern protocols are based on the Open System Interconnection model, in which interoperability between network devices and software is broken down into layers. In the FPGA world, intellectual property (IP) in libraries such as the Xilinx LogiCORE™ and AllianceCORE often uses higher-level serial connectivity protocols such as PCI Express® in addition to lower-level physical-layer (PHY) protocols like 1000BASE-X.

However, determining the correct PHY protocol template for your given design project is not always as straightforward as selecting a higher-level protocol. As with many industries, consolidation and design reuse can create a complex maze that you must navigate. Understanding the "higher" layer protocol and its relationship to the "lower" layer protocol specifications—and being mindful of how each industry defines the PHY—will help you choose the best Xilinx LogiCORE IP High-Speed Serial Transceiver Architecture Wizard protocol template to achieve your design goals (*www.xilinx.com/products/design_resources/conn_central/solution_kits/wizards/index.htm*).

From highest to lowest, the OSI ladder consists of the application, presentation, session, transport, network, data link and physical layers.

Protocols of the application layer directly serve the end user by distributing information services appropriate to an application, to its management and to system management. The next-highest rung in the OSI model, the presentation layer, provides a set of services that the application layer may select to enable it to interpret the meaning of the data exchanged. These services are for the management of the entry exchange, display and control of structured data.

establish, maintain and release physical connections between data link entities.

### Three PHY Sublayers

Many of today's popular serial connectivity protocols mimic the OSI layers model and use similar terminology, for example denoting lower to upper layers as Nos. 1 through 7 in a hierarchy ranging from physical to application layers. For the purpose of this article, when referring to a serial connectivity protocol the term "higher" refers to Layers 2-4 (or those above the physical layer).

The PHY layer (Layer 1) comprises two to three sublayers: the physical coding sublayer (PCS), the physical-medium

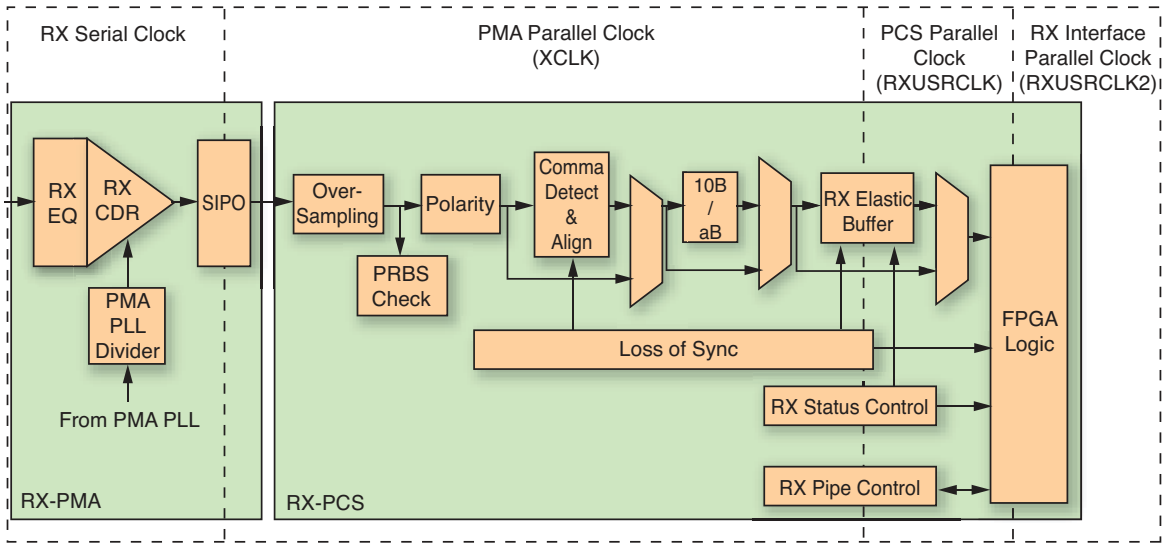



*Figure 1 – Block diagram of example Virtex-5 RX physical sublayers PCS, PMA and PMD*

Let's review these protocols and then take a look at the best methods to help you select the right one for your design.

### OSI: A Template for Connectivity Protocols

The Open System Interconnection, or OSI, is an ISO standard for worldwide communications (*http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=20269*) that defines a framework for implementing protocols in seven layers. Control passes from one layer to the next, starting at the application layer in one station and proceeding to the bottom, physical layer, over the channel to the next station and back up the hierarchy.

The session layer assists in supporting the interactions between cooperating presentation entities, while the transport layer provides a universal transport service in association with the underlying services that the lower layers supply. The network layer, for its part, provides functional and procedural means to exchange network service data units between two transport entities over a network connection.

Finally, the data link layer provides the functionality and procedural means to establish, maintain and release data links between network entities, while the physical layer offers mechanical, electrical, functional and procedural characteristics to

attachment (PMA) sublayer and a third, optional, sublayer called physical-medium dependent (PMD). When referring to the PHY layer, we use "higher" in reference to PCS and "lower" for the PMA and PMD layers. Figure 1 depicts the layers in a block diagram.

When transmitting, the packets or data travel in forward order: media-access control (MAC) layer to PCS, PMA and PMD; they go in reverse order when receiving. How Xilinx implements the PHY sublayers in the high-speed serial transceiver will depend upon the protocol,

The physical coding sublayer interfaces to the higher-level Layer 2 data link (or

**You can implement the physical layer, which designers often refer to as the electrical specification, in a single or multiple devices. The PHY sublayer usage will depend on the market segment and protocol.**

MAC) layer. It typically implements 8b/10b encoding/decoding, comma alignment, channel bonding and clock correction.

Higher-layer protocol specifications may define the PCS as part of the PHY or refer to an industry-standard PCS. For example, the second-generation Serial RapidIO specification defines the PCS but refers designers to use the CEI-6G-SR/LR specification for the PMA. You can implement the PCS in the high-speed serial transceiver, in the body of the FPGA or in a combination of the two.

The physical-media attachment sublayer, meanwhile, is often referred to as the "electrical specification." The PMA implements the appropriate signal integrity of the protocol as well as a number of other typical features. They include current-mode logic (CML) drivers/buffers with configurable termination, voltage swing and coupling; programmable transmit pre-emphasis and receive equalization for optimal signal integrity; and line rates depending on device and transceiver type (for example, Spartan®-6 GTP, Virtex®-6 GTX) with optional oversampling.

Additional functions the PMA handles include fixed latency modes for minimized, deterministic datapath latency; out-of-band signaling support (specifically designed to address the requirements of PCI Express and Serial ATA protocols; and built-in pseudo-random bitstream generation/checking logic for easier bit-error-rate checking.

Finally, the physical-medium dependent sublayer is an optional part of the PHY layer specifications generally used with Ethernet protocols. The PMD is responsible for the transmission and reception of individual bits on a physical medium. Its jobs encompass bit timing, signal encoding and interacting with the physical medium and the cable or the wire.

In implementing a 1000BASE-X PCS/PMA LogiCORE, for example, the 1-Gigabit Ethernet MAC connects to the LogiCORE, which in turn connects to a 1000BASE-X PMD optical transceiver. The Xilinx Trimode Ethernet MAC (TEMAC) LogiCORE implements the PCS and PMA functions—the latter in the high-speed serial transceiver, sometimes called a gigabit transceiver (GT). In the PCS layer, the high-speed serial transceiver implements the encoding and decoding, while the FPGA logic handles autonegotiation. The PMD sublayer contains a transceiver for the physical medium.

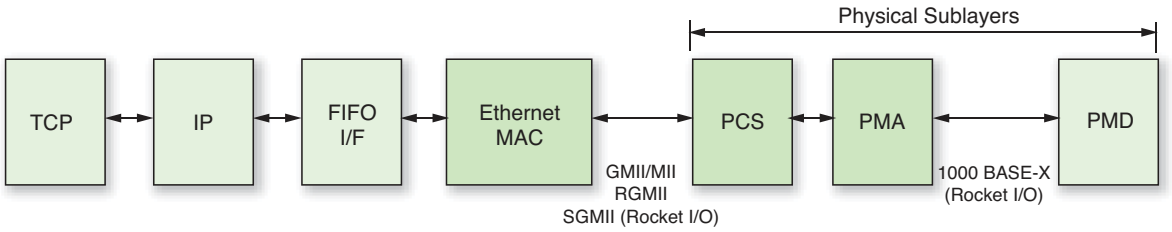



*Figure 2 – Example of PHY PCS, PMA and PMD layers in an Ethernet communications application*

### Confusion Over PHY Usage

It's easy to get confused over the usage of PHY as a silicon chip. The PHY is a specification layer consisting of sublayers. You can implement the PHY, which designers often refer to as the electrical specification, in a single or multiple devices. The sublayer usage depends on the market segment and protocol. In the case of the popular serial protocol PCI Express, the PHY consist of PCS and PMA sublayers.

The PHY layers in communication protocols commonly use the PCS, PMA and PMD sublayers. Figure 2 is an example of the use of the Xilinx TEMAC (10M/100M/1G) LogiCORE in a local-area network application where the 1-Gbit Ethernet MAC communicates to a 1000BASE-X PCS/PMA and to a laser optical-transceiver 1000BASE-X PMD. Here, the PHY is implemented in both the FPGA and in an optional optical-transceiver device.

Because PHY layers can be confusing, it is best to read the specification so that you can understand exactly what function the layers will perform (that is, PCS, PMA, PMD) and where the function is implemented in the silicon (FPGA logic body, high-speed serial transceiver or outside of the FPGA). In most cases the PMA is assumed to be part of the PMD.

Of course, there are many connectivity protocols (for example, PCI) that are not serial (that is, single-ended or differential-signal I/O) and do not use a high-speed serial transceiver. In some cases you may implement the serial PHY using the SelectIO™ serializer/deserializer (serdes).

### Hardened or Embedded IP Considerations

Often, Xilinx will put popular ubiquitous protocols like PCI Express and Gigabit Ethernet right into the FPGA. These "hardened" versions may implement all or part of the protocol. In these two cases, the LogiCORE wrapper implements the MAC and physical layer (PCS and PMA) as part of the LogiCORE product. The wrapper includes the hardened block and connects it with the high-speed serial transceiver. In the case of the TEMAC, the hardened IP implements the MAC and portions of the PCS, along with the transaction and data link layers of the PCI Express LogiCORE. You can use Xilinx's High-speed Serial Transceiver Wizard to view and modify the GTP/GTX settings. (For details, refer to the documentation for the Tri-mode Ethernet Wrapper for Integrated Block at *www.xilinx.com/products/design_resources/conn_central/protocols/gigabit_ethernet.htm*; and for the Endpoint BlockPlus for PCI Express LogiCORE at *www.xilinx.com/products/design_resources/conn_central/solution_kits/pci/index.htm*.)

| |
|---|
| Aurora 8b/10b, Aurora 64b/66b |
| DisplayPort |
| GPON |
| Interlaken |
| OC-48 |
| PCI Express Gen1/Gen2 |
| Serial RapidIO Gen1 |
| HD-SDI |

*Table 1 – Higher-layer protocol specification defining PCS/PMA*

| Higher protocol | PCS/PMA protocol |
|---|---|
| 10G Ethernet (XFI/SFI) | 10GBASE-X, XAUI |
| CPRI | 1000BASE-CX, XAUI, LVXAUI |
| OBSAI | CEI-6G-SR/LR and XAUI |

*Table 2 – Higher-layer protocol defining the use of PCS and PMA specifications*

| Lower PCS/PMA protocol | Higher protocol |
|---|---|
| 1000BASE-X | Xilinx Trimode Ethernet MAC (TEMAC) |
| CEI-6G-SR/LR | OBSAI RF03-01 LVXAUI |
| SGMII | Xilinx TEMAC |
| XAUI | 10GE |

*Table 3 – Lower-layer protocol defining both the PCS (or portion of PCS) and electrical PMA (or portion of PMA)*

### Wizard Protocol List

Tables 1, 2 and 3 provide lists of the serial protocols the High-speed Serial Transceiver Wizards support. Table 1 contains a list of higher-layer protocols whose specification includes the PCS and PMA, while Table 2 consists of higher-layer protocol specifications that define the usage of an industry-standard PCS and/or PMA specification. Table 3 lists physical-layer specifications and the higher-level protocols commonly used with them.

To best help you customize your PHY using the GT architecture wizard, let's take a more-detailed look at the PHY layers of the top serial protocols and their corresponding wizard protocol templates. The most commonly used protocols include 10G Ethernet MAC—XAUI, CPRI v4.0, 3G and 6G OBSAI RP3-01, first- and second-generation PCI Express, Serial RapidIO, HD-SDI and Xilinx TEMAC (10M/100M/1G Ethernet).

Before using the GT wizard, I advise new users to become familiar with the Virtex-6 or Spartan-6 GT datasheet, each of which contains lists of supported protocols. Also, while using the wizard is fairly straightforward, it isn't a bad idea to review the "Wizard Getting Started Guides" for both of those devices, which contain a lot of useful information (see *www.xilinx.com/products/design_resources/conn_central/solution_kits/wizards/3*).

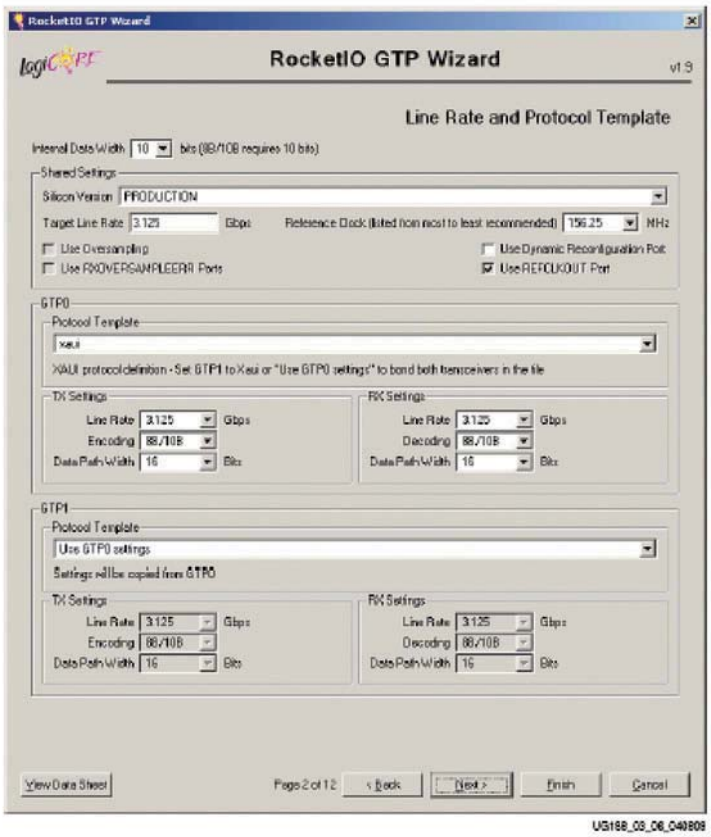



*Figure 3 – Virtex-6 GTP Wizard walks users through the high-speed I/O setup.*

*Table 4 – Protocol templates for Virtex-6 and Spartan-6 gigabit transceiver wizards*

| GT Wizard Protocol Template | Virtex-6 GTX | Spartan-6 GTP | Description | Protocol Specification |
|---|---|---|---|---|
| 3G OBSAI | ★ | ★ | Open Base Station Architecture Initiative. OBSAI RF03 is a subset of RF3-01. *www.xilinx.com/esp/wireless.htm* | Open Base Station Architecture Initiative. *www.obsai.org*; Optical Internetworking Forum (OIF), *www.oiforum.com*; XAUI (IEEE), *www.ieee.org* |
| 6G OBSAI | ★ | | Open Base Station Architecture Initiative. OBSAI RF03 is a subset of RF3-01 *www.xilinx.com/esp/wireless.htm* | Open Base Station Architecture Initiative, *www.obsai.org*; Optical Internetworking Forum (OIF), *www.oiforum.com* |
| Aurora 8b/10b | ★ | Due soon | Lightweight link layer | Xilinx, *www.xilinx.com/aurora* |
| Aurora 64b/66b | ★ | | Lightweight link layer | Xilinx, *www.xilinx.com/aurora* |
| CPRI v4.0 | ★ | ★ | *www.xilinx.com/esp/wireless.htm* | Common Public Radio Interface, *www.cpri.info* |
| DisplayPort | Due soon | ★ | Display protocol, *www.xilinx.com/products/design_resources/conn_central/grouping/displayport.htm* | Video Electronic Standards Assoication (VESA), *www.vesa.org, www.displayport.org* |
| Fibre Channel | ★ | | PHY is similar to XAUI's. *www.xilinx.com/products/design_resources/conn_central/grouping/fibre_channel.htm* | Fibre Channel Industry Association (FCIA), *www.fibrechannel.org* |
| GigE (SGMII/1000Base-X) | ★ | ★ | Support by Xilinx TEMAC. *www.xilinx.com/ethernet, www.xilinx.com/esp/wired.htm* | IEEE, *www.ieee.org* |
| GPON | ★ | | Fiber to the home. Gigabit passive optical network. *www.xilinx.com/esp/wired.htm* | International Telecommunication Union, *www.itu.int* |
| HD-SDI | ★ | ★ | High definition. One rate, 1.485G, used in broadcast. *www.xilinx.com/esp/broadcast.htm* | Society of Motion Picture and Television Engineers (SMPTE), *www.smpte.org/home* |
| Interlaken | ★ | | Flexible chip-to-chip packet transfers. Works with many MACs: OC768 Sonet, 100GE. *www.xilinx.com/esp/wired.htm* | *www.interlakenalliance.com* |
| OTN OTU2 (XFI) | | Due soon | *www.xilinx.com/esp/wired.htm* | XFI electrical interface specification is a part of XFP Multi Source Agreement Specification. *www.xfpmsa.com/cgi-bin/msa.cgi* |
| PCI Express Gen1 | ★ | ★ | First-generation PCI Express. *www.xilinx.com/pciexpress* | PCI-SIG, *www.pcisig.org* |
| PCI Express Gen2 | ★ | | Second-generation PCI Express. *www.xilinx.com/pciexpress* | PCI-SIG, *www.pcisig.org* |
| Serial RapidIO Gen1 | ★ | ★ | First-generation Serial RapidIO. *www.xilinx.com/esp/wireless.htm* | RapidIO, *www.rapidio.org* |
| TEMAC using SGMII | ★ | ★ | Support by Xilinx TEMAC. *www.xilinx.com/ethernet, www.xilinx.com/esp/wired.htm* | IEEE, *www.ieee.org* |
| XAUI | ★ | TBD | 10G Ethernet Extended Attachment Unit Interface (XAUI). Connects to 10GE MAC. *www.xilinx.com/ethernet, www.xilinx.com/esp/wired.htm* | 10GE & XAUI: IEEE, *www.ieee.org* |

Figure 3 shows the Virtex-6 GTX wizard GUI, with a pull-down menu for the protocol templates. Table 4, meanwhile, contains a list of protocol templates that the Virtex-6 GTX and Spartan-6 GTP wizard LogiCOREs support.

At the request of customers, Xilinx has improved the use model between the LogiCOREs and the GT wizards. Many Xilinx serial LogiCOREs—including those for Serial RapidIO, XAUI and Aurora—now use the output of the wizards directly. Customers can select a protocol template like Serial RapidIO and make custom modifications. System I/O specialists no longer need to work with the LogiCORE engineering team to modify the serdes setting to create custom protocols.

### 10G Ethernet - XAUI

The 10-Gigabit Ethernet (also known as 10GbE, 10GigE or 10GE) standard is an IEEE specification that defines a nominal rate 10 times as fast as Gigabit Ethernet. The physical layer consists of an interface from the MAC to the PHY, PCS, PMA and PMD. In the case of the Xilinx LogiCORE, the 10-Gbit Media-Independent Interface (XGMII) can connect to an optical module or to the 10G Ethernet Extended Attachment Unit Interface, better known as XAUI. The PMA and PMD are either an external device, as in the example of an optical transceiver, or considered part of the XAUI, as in the case of a chip-to-chip or backplane application.

The Virtex-6 LXT, SXT and HXT FPGAs do not support a one-lane, 10G data transfer rate to interface to a single-lane optical-fiber PMD. XAUI's four lanes are used to interface to the optical PHY module, which is then able to send/receive the 10GE packets to their destination through a single fiber cable.

Designers supporting applications in many markets, including networking and telecommunications, commonly use XAUI to connect board-to-board over backplanes, or to connect to 10-Gbit optical modules. Used in a variety of higher-level protocols, XAUI can also serve as a chip-to-chip MAC/PHY interface, since the 72-pin XGMII is not a practical one to use on a circuit board header or off-board interface.

A designer can use the Xilinx XAUI LogiCORE to implement the DTE XGXS (another acronym for XAUI, it stands for "10-Gbit extender sublayer"), PHY XGXS and 10G BASE-X PCS using both the gigabit transceiver and FPGA logic. The XAUI interface is considered a PCS layer that can interface to a 10G Ethernet PHY on one side and the MAC on the other. The FPGA would not implement the 10G Ethernet PHY, which could be optical or copper.

Xilinx customized the GT wizard's 10GE XAUI template for the Xilinx 10GE XAUI LogiCORE. For custom applications or AllianceCORE, refer to the IP documentation implementation details. The designer can use the GT wizard to define the electrical attributes of the gigabit transceiver for a XAUI PCS.

### Common Packet Radio Interface v4.0

You can use the Common Packet Radio Interface (CPRI) for connectivity between radio equipment controllers or base stations and one or more radio equipment units. The specification covers Layers 1 and 2 of the OSI stack, with the physical layer (Layer 1) defining both the electrical interface used in traditional base stations and an optical interface for base stations with remote radio equipment. The Xilinx CPRI LogiCORE implements the PHY in the GT and the data link (Layer 2) in the logic body of the FPGA.

CPRI specifies line bit rates of 614.4 Mbits/second (E6), 1,228.8 Mbits/s (E12) and 2,457.6 Mbits/s (E24). It specifies both high-voltage and low-voltage variants for E6 and E12, whereas E24 has only a low-voltage specification. The high-voltage variants are based on IEEE 802.3-2002, Clause 39 (1000BASE-CX) and the LV variant on IEEE 802.3ae-2002, Clause 47 (XAUI). CPRI can also use the low-voltage XAUI specification, LVXAUI.

You can use the GT wizard to define the electrical attributes of the GT as per the standard specifications CPRI v4.0 for the Xilinx CPRI v4.0 LogiCORE, which uses a modified version of 100BASE-CS and XAUI for the electrical. Use the menu item "CPRI v4.0" when viewing or modifying the Xilinx CPRI v4.0 LogiCORE.

### 3G and 6G OBSAI RP3-01

The architecture of the Open Base Station Architecture Initiative (OBSAI) RP3-01 cellular base station protocol is divided into the lower physical layer and the higher application, transport and data link layers. The application layer can interface with baseband or RF cards, while the data link layer interfaces with the physical layer. Xilinx implements the PHY using a transceiver in the FPGA to handle the electrical portion and connecting it to an external optical-transceiver module.

In the case of OBSAI, the 8b/10b encoding and word-alignment blocks, as well as the synchronization and phase-alignment buffers, are embedded in the transceiver blocks and do not require the use of FPGA logic.

The OBSAI RP3-01 electrical parameters comply with XAUI electrical (Clause 47 of IEEE 802.3ae-2002) up to 3.072 Gbits/s and Common Electrical IO (CEI) G6 for both the short-reach (CEI-6G-SR) and long-reach (CEI-6G-LR) 6.25-Gbit/s standards. The GT wizard provides protocol templates for the Xilinx 3G OBSAI RP3-01 and 6G OBSAI RP3-01 LogiCORE, implemented using a modified CEI-6G, XAUI/SRIO electrical. The menu items are 3G OBSAI RP3-01 and 6G OBSAI RP3-01.

### PCE Express, Generations 1 and 2

The PCI Express protocol is defined in three layers: physical, data link and transaction. Thanks to its popularity, newer serial protocols may choose to be electrically compatible with or similar to PCI Express, allowing vendors of ASSPs and other PHY devices to reuse portions of their well-tested IP portfolios. Xilinx implements the first- and second-generation PCI Express protocols in an integrated hard-IP block (Endpoint Block Plus Wrapper for PCI Express LogiCORE) and as soft IP through Xilinx and AllianceCORE partners.

You can use the GT wizard to generate the settings for customization to be manually migrated into the source code wrapper of the hard-IP block. The wizard has two protocol templates to support the PCI Express physical layer: PCI Express Gen1 and PCI Express Gen2.

### Serial RapidIO

Like PCI Express, the Serial RapidIO protocol is also defined in three layers, in this case, physical, logical and transport. It is the nature of this industry to leverage as much of existing technical specifications as is relevant for the purpose of the protocol. Thus, Serial RapidIO's physical layer uses the XAUI electrical interface (IEEE 802.3ae-2002, Clause 47) as a guide in defining its own parameters for the AC electrical specification. Since RapidIO and XAUI have similar applications goals, Serial RapidIO designers were able to reuse their existing XAUI electrical designs. The GT wizard supports the Serial RapidIO PHY by means of the Serial RapidIO template.

### Triple-Rate SDI Video

Triple-rate SDI video reference designs are based upon the Society of Motion Picture and Television Engineers (SMPTE) standards. Our integrated reference design supports several standard- and high-definition flavors of this standard, namely SD-SDI, HD-SDI, Dual Link HD-SDI and 3G-SDI.

The physical connection to the High-Speed Serial Transceiver is differential CML driving an external cable driver (for transmit) or an external adaptive receiver equalizer. The common serialization protocol between the standards is very specific and is designed in the FPGA fabric. This protocol has long runs of 1's and 0's that require very large AC coupling capacitors.

Broadcast equipment is networked together with the Triple-Rate SDI standards running over 75-ohm coaxial cables. The GT wizard supports SD/HD/3G-SDI reference designs using the HD-SDI protocol with the template of that name. The tri-mode reference design uses the dynamic-reconfiguration port to reconfigure the gigabit transceiver to support the SD and 3G trimodes. In these modes, SD-SDI and 3G-SDI use the phase-locked loop inside the GT to recover the data at 2.97 Gbits/s.

### Xilinx Trimode Ethernet

The Trimode Ethernet MAC is a Xilinx implementation of the 10/100/1G Ethernet protocol. Xilinx offers the TEMAC LogiCORE (soft IP) and Trimode Ethernet Wrapper for Integrated Block (hard IP). For the soft IP, the 1000BASE-X PCS/PMA or SGMII LogiCORE can connect seamlessly. SGMII is a serial connectivity standard that supports 10/100/1G operation.

The connection between the TEMAC and the GTX is always 1 Gbit/s, but the interface will sample less frequently for the slower rates. The SGMII option is different in the wizard's PCS and PMA settings, and is available as a separate protocol template.

Applications using copper wires (1000BASE-T) do not have the same division of logic between the FPGA and the 1000BASE-T PHY as in a 1000BASE-X ("X" signifies optical) PMD device. In the case of a 1000BASE-T device, the PCS, the PHY device generally implements the PMA and PMD PHY sublayers. The designer can implement the MAC in the FPGA and the GT through a GMII interface; the MAC does not to implement any of the PHY layer. Thus, there is no GT wizard protocol template for 1000BASE-T.

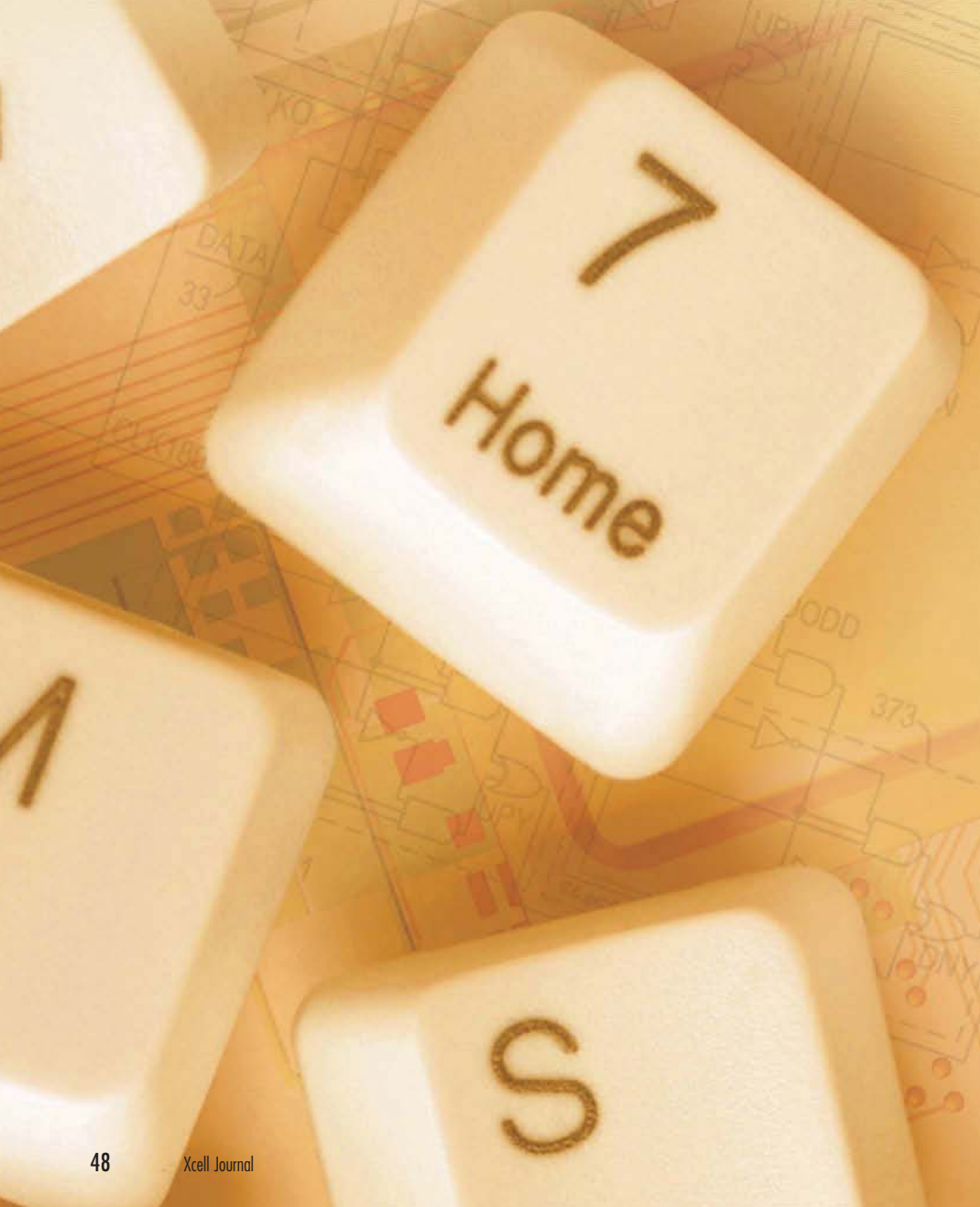
The TEMAC wrapper is an HDL wrapper around the hard TEMAC sub-block and the GT I/O blocks (1000BASE-X/SGMII is integrated into the hard TEMAC already). The Ethernet LogiCORE documentation supplies implementation details.

The GT wizard supports the Tri-Mode Ethernet protocol using the GigE (SGMII/1000Base-X) template.

In conclusion, industry-standard protocols are constantly changing, and it seems one or two new ones emerge every year. As such, the terminology and underlying technologies can be as confusing as tax law. The more you understand about the details of a given protocol's physical-layer scheme, the easier it will be to determine the best high-speed serial transceiver wizard protocol template to use as a starting point for your design projects.

# FPGA Anchors Innovative General-Purpose PC

## A rethinking of 41-year-old microprocessor and software-OS model yields a new architecture that's fast, reliable and immune to viruses.




by Gregg J. Macdonald
Design Engineer
Electronic Compute Systems, Inc.
*Gregg.Macdonald@ecs-pc.com*

Personal computers are all around us, helping us in innumerable ways and enhancing our productivity. In fact, so ubiquitous is the PC that many people now take it for granted. But since the debut of the microprocessor in 1968, design engineers collectively have yet to compress all of the things we want the PC to do into one small, easy-to-manage "can-really-do-anything" box.

For many common tasks these computers perform very well. But for many others, they are too slow or unreliable. In some cases we can't even trust the computer will turn on or be ready when we are. Sometimes they hang. Other times they slow to a crawl. There are well-known fixes for most of these problems, but even a top-of-the-line PC can't smoothly stream from the Web, play music, show a video and TV show, and process a telephone call simultaneously, as a spreadsheet runs in the background. The jobs it can do will often have lots of dependencies, a fact that points to the PC's Achilles' heel.

Like it or not, this time-multiplexed serial fetch-decode-execute architecture has not fundamentally changed in 41 years. A consequence of that architecture is that the PC performs hardly any tasks concurrently, except in the case of recent multicore machines. But even in these PCs, another software operating system must sit on top of the multiple processor cores, using the same time-multiplexed serial fetch-decode-execute architecture. Piling on more and more microprocessors per chip creates complexity. Such an architecture consumes lots of concurrent clock cycles fetching, decoding and executing instructions that may simply result in NOP (no operation) or in getting another instruction.

A new approach using a Xilinx® FPGA as the core of the personal computer can solve these problems, providing a tremendous breakthrough for the full spectrum of old and new embedded applications. I devised this approach—which I call HUMBLE, for Hardware Unified Multiple Branch Logic Engine—over the course of 10 years' work as an antidote to the existing PC architecture. I call the latter SUSBLE, for Software Unified Single Branch Logic Engine. (For their part, multicore machines might be termed SUMBLEs: Software Unified Multiple Branch Logic Engines.)

SUSBLE arises from the Von Neumann-Harvard fetch-decode-execute architecture, which is further subdivided into RISC and CISC (reduced or complex instruction-set computing, respectively). So while both types of SUSBLE machines are instruction-set computers, HUMBLE is not. Its architecture could thus be called NISC. Instead of spending time fetching instructions, HUMBLE springs into action immediately, simply because its foundation is Xilinx logic cells, slices and cores, which themselves are essentially always ready to "do" things.

### Instant On

SUSBLE computers typically require 15 seconds to 2 minutes of checking and setting things up immediately upon turn-on, with many dependencies. This offers a hint at how long consecutive strings of SUSBLE processes really take before you can do something useful with the mouse or strike a key. In contrast, the HUMBLE PC with a Xilinx FPGA as the center-stage core turns on in approximately 1 second, and it seems reasonable to reduce this further.

*Figure 1 – The HUMBLE PC turns on in a second, thanks to a microprocessor-less architecture that has no need to fetch instructions.*

By definition, HUMBLE doesn't need to fetch instructions. When using this computer, there is no need to start a microprocessor, fetch boot code, set up a cache, maintain cache coherency and so on. Xilinx logic cells, slices and cores are born ready. By nature they are "instant on" (after configuration). Thus, configuration time for the HUMBLE PC motherboard (also known as the Xilinx ML401-VLX25) in serial mode at 40 MHz works out to approximately 0.2 second. Parallel mode is certainly faster, but our design goal of 1-second readiness at the user level is easily achieved.

### Faster at 1/10 Clocking Speed

Whereas HUMBLE is ready in approximately 1 second, my XP machine, built around a 1.2-GHz Celeron processor with 384 Mbytes of RAM, sometimes takes as long as 2 minutes to turn on. To be fair, I don't really know everything XP is doing in that time. However, I observe that it takes approximately 60 seconds before I can begin to click on anything with the mouse and have any effect. For an additional 75 seconds the light on the hard disk continues to flicker. It doesn't stop until approximately 135 total seconds have elapsed. For the sake of argument, let's say 120 seconds, because it's a nice round number and because we know the PC is still doing something relating to readiness; so 120/1 second = 120.

The HUMBLE PC, meanwhile, is running a digital clock manager (DCM) at 100 MHz, so that means it's operating at 1/12 the clocking frequency of the XP machine. If I normalize the clock frequencies with respect to the 1/10 claim, I must multiply 120 by 1.2, giving us a factor of 144 in favor of HUMBLE. In other words, HUMBLE with Xilinx Virtex® onboard achieves my design goal 144 times faster.

Certainly 1.2 GHz is no longer state of the art in conventional PCs; newer Celeron processors operate at 2 GHz or more (not to mention duals and quads), scaling the performance while typically still running a lot of character-per-second applications. So as a second example, let's compare HUMBLE with a new single-core Vista machine. With the Vista PC, equipped with a 2.4-GHz Celeron and 2 Gbytes of RAM, it takes 44 seconds before I can do anything with the mouse or keyboard, though it has no light to monitor what activity may or

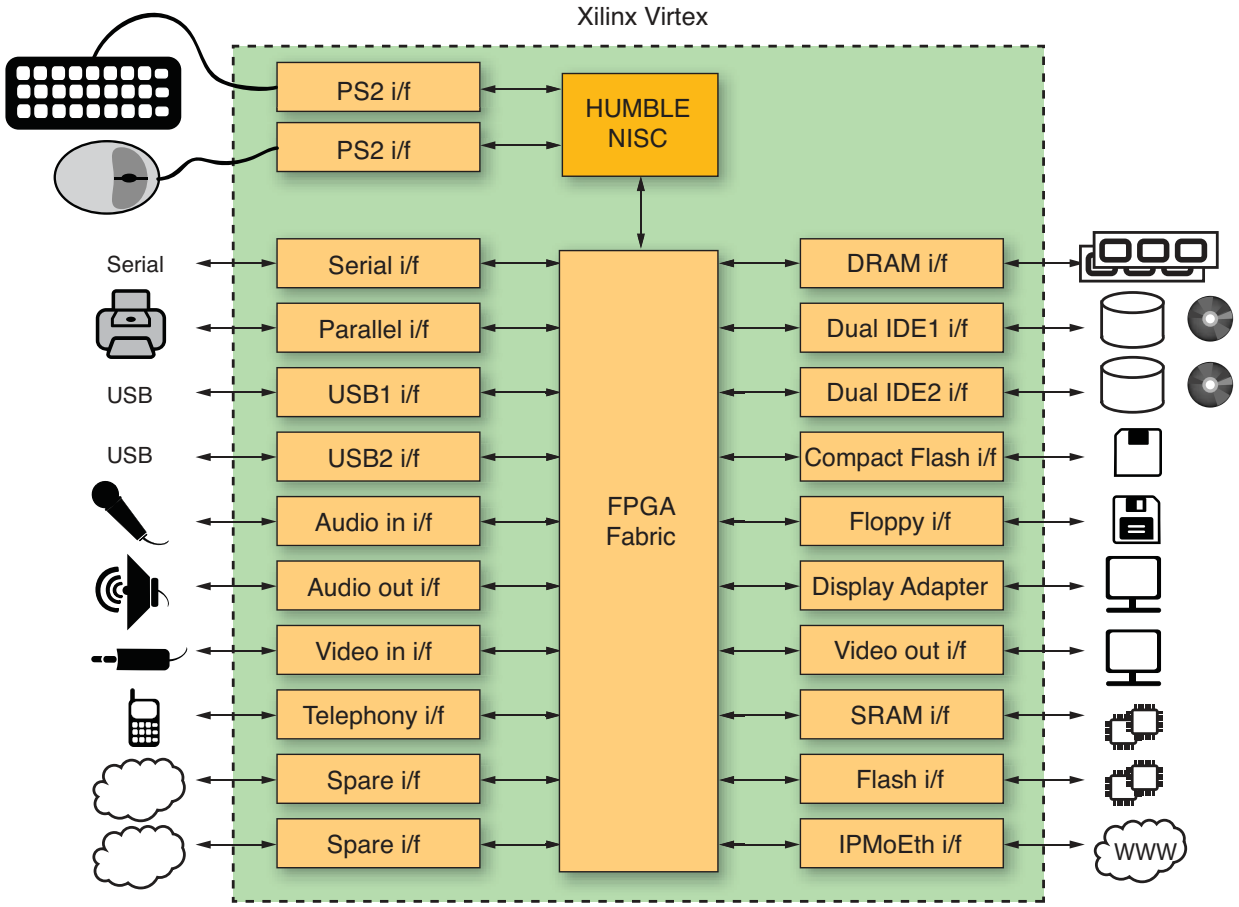



*Figure 2 – Block diagram of HUMBLE PC built around Xilinx Virtex FPGA*

may not continue to and from the SATA hard drive. Using that number (44) and normalizing the 2.4-GHz processor speed to HUMBLE's 100 MHz gives us a 2.4 multiplier with respect to my 1/10 clocking-speed claim. So, 44 times 2.4 = ~106.

The new HUMBLE approach installs Xilinx logic cells, slices and cores in a parallel manner to provide concurrency, resulting in many multiples of execute-execute-execute at much lower clock frequency than a conventional PC architecture and correspondingly lower power. To say this another way, there simply is no need for gigahertz clocking to perform typical PC functions, including character-per-second applications. For other kinds of programs that need maximum algorithmic acceleration, including embedded circuits and applications greater than character per second, something closer to whatever makes sense is possible within a HUMBLE PC.

The HUMBLE PC uses Xilinx DCMs and phase-matched clock dividers (PMCDs). To support the file system and just about any kind of anticipated user application, it also includes a time, date, calendar and time zone hardware program using Xilinx logic cells, slices and cores. For embedded applications, HUMBLE with Xilinx DCMs and PMCDs provides up- and down-frequency selectivity and the following standard user clocks: 200, 133, 100, 66, 50, 33, 25, 16.5 and 12.5 MHz; 1, 4 and 100 microseconds; 10 milliseconds; 1/4, 1/2, 1 second, 1 minute, 1 hour, 1 day, 1 week, 1 month and 1 year. There are also several spare DCMs and PMCDs for generating any other desired frequency. Clearly, there is no gigahertz speed going on here, but if an embedded application needs it, Xilinx Virtex parts provide this too.

### Lower Power

Using Xilinx parts in combination with the HUMBLE architecture as the engine, there is no need to include a cache, cache controller, microprocessor or soft microprocessor to fetch, decode and wait the 10 to 100 cycles or more per instruction for execution. This is one very basic way we achieve lower power. Further, for the HUMBLE PC motherboard, we chose the Xilinx ML401. This board allows HUMBLE to integrate the display adapter (aka graphics adapter) and motherboard clock chip. Historically, neither of these has been part of the microprocessor. Then we use Xilinx logic cells, slices and cores again in combi-

We remove the OS from the microprocessor—in fact, we go a step further and remove the whole microprocessor. Instead, HUMBLE loads the OS at turn-on from write-protected flash memory as a 'hardware' program.

nation with the HUMBLE architecture to describe the display adapter, enabling further power reduction. Other methods for curbing power consumption are also available with the Xilinx Virtex series, such as turning off the clock or regions of clocks.

By nature, Xilinx FPGAs have always been popular because they can solve a plethora of real-time problems. HUMBLE leverages this capability, intending to make things easier wherever possible. If nothing else, HUMBLE and Xilinx allow just about any real-time application to coexist on the same silicon. Hence, HUMBLE isn't just a breakthrough for PCs, it is a breakthrough for all the embedded applications that today's microprocessors can't do—or can't do well.

### Immune to Viruses

In addition, the HUMBLE architecture is immune to viruses in ways that existing microprocessor-based computers cannot be. In SUSBLE PCs, the operating system and programs are not write-protected, nor can they be with any benefit thereof. When any program is running, it has access to and control of nearly the entire machine and can do whatever the author has intended. Before any program, including the OS, begins to run, it must start from somewhere (aka boot). Thus, a "boot virus" can do whatever it wants to, including disabling the OS from blocking any intended malicious actions, thereby allowing control of the entire machine without exception. A simple virus program can destroy or change any part of the operating system, application programs or user data, because programs (on disk) all appear the same to a running program as read/writable objects (data). In short, with SUSBLE computers, viruses will remain a perpetual problem because there is no real root solution possible.

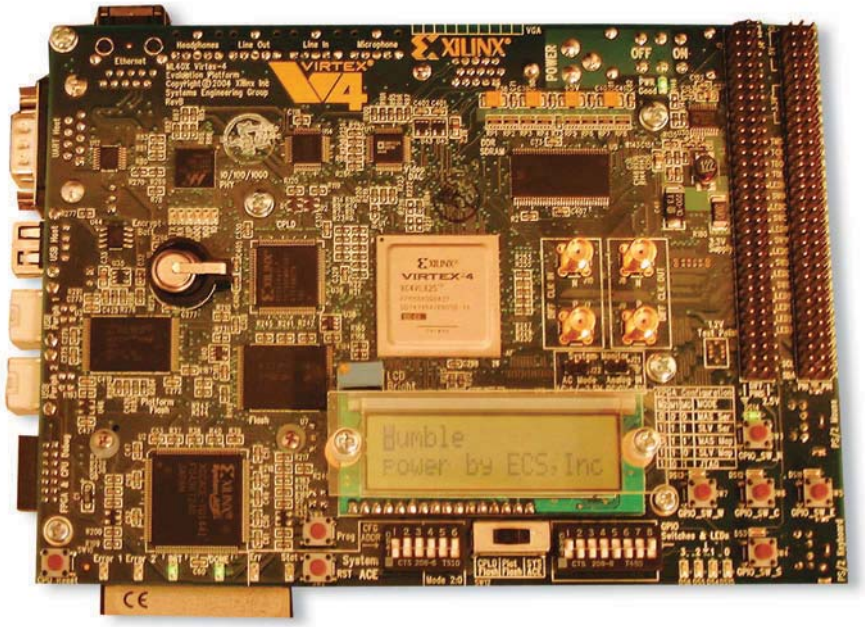



*Figure 3 – The ML401 board lets the HUMBLE PC integrate the display adapter (aka graphics adapter) and motherboard clock chip.*

In the case of HUMBLE, we remove the OS from the microprocessor—in fact, we go a step further and remove the whole microprocessor. HUMBLE doesn't load or fetch or architect the OS as a running software program. Although it loads at turn-on, it does so from nonremovable write-protected flash memory as a "hardware" program. This protected nonremovable media is not associated with user media and drives or ports. Saying this another way, while HUMBLE loads its personality upon turn-on, it doesn't load it from a floppy or hard disk or other unprotected read/write media. Programs cannot access HUMBLE's fixed protected flash memory (on the motherboard) during runtime; the memory is not writable by itself or by other running programs.

Presently, the Xilinx Parallel IV cable programming interface is required for changes. This method allows upgrades to the operating system while preventing virus entry altogether into the OS. So, by intentional design it is impossible for a user program (virus) to disable or overwrite any part or portion of the on-chip silicon-based-operating system or silicon-based programs.

Certainly it can be argued that virus entry is feasible through the Xilinx programming interface. While that is theoretically possible, in fact this interface typically will be used once or a small number of times, usually at the manufacturer or an affiliate. So clearly, this is not a wide-open point of entry (as is the case with present microprocessor-based computers). A virus would have to come directly from the manufacturer or affiliate, hence reducing the probability of occurrence. The problem becomes one of quality control on the part of a manufacturer. So while viruses are not 100 percent impossible, the chance of occurrence is tremendously reduced—we suggest by as much as 99.9 percent.

Even if a virus were to enter the system in this way, however, it could not spread to

**Xilinx Virtex FPGAs deliver millions of instant-on reconfigurable and parallelable transistors on a chip. The HUMBLE approach constructs a computing engine that exploits this parallelism.**

other HUMBLE computers; it would compromise only the one affected machine. So the virus isn't contagious, it isn't virulent. In fact, it is self-limiting—self-quarantined—at least among other interconnected HUMBLE computers. Assuming a case of virus entry, the solution is simple: just revert to the previous known-good version of the operating system or delete the affected user data.

### Extending Legacy Programs

The present HUMBLE PC and development system do not support existing x86 programs. This is intentional, to make the HUMBLE innovation contained within the open architecture of a Xilinx Virtex FPGA—and corresponding open architecture of the Xilinx ML401 board—clear to all. Having said this, many options are available to bridge this new architecture back to x86—for example, through the file system, Ethernet connection, display adapter (frame buffer) or other creative means.

The reprogrammability of FPGAs conjures the possibility of selectively configuring HUMBLE or SUSBLE, or some combination of both, at turn-on within the same Xilinx motherboard. As another example, HUMBLE could serve as the supervisory operating system on top of a given connection of gigahertz (or less) CISC microprocessors—hard or soft, duals, quads and so on. This seems to make sense because of the architecture's natural concurrency without need for an instruction stream. Presently, Electronic Compute Systems, Inc. has no effort under way to bridge HUMBLE to x86. So this is a wide-open opportunity for third parties.

### How HUMBLE Parallels Xilinx Logic Cells

The idea of using parallel logic or paralleling logic doesn't make much sense from a conventional software perspective, because it usually cannot be performed except in cases where multiples of the actual hardware elements—components, functions and so on—exist, and the software compiler allows access. Even then it can be quite obtuse, given the nature of software-driven fetch-decode-execute architectures. Hence, paralleling logic is almost completely foreign when writing software.

Historically, software engineering or at least a mind-set toward software is necessary when in the driver's seat as a system architect, designer or product engineer. This legacy is wise and makes good sense for most companies, because most systems are microprocessor based. Hence we are presently living in a mostly software-driven (serial-time-multiplexed) world of products.

However, Xilinx Virtex FPGAs now deliver millions of instant-on reconfigurable and parallelable (in almost any combination) transistors on a chip. Unsurprisingly, they are often still used in a serial-time-multiplexed fashion without exploiting their additional feature of inherent parallelism. For example, there are approximately 11,000 slices (24,000 logic cells) in the Xilinx VLX 25, all on the same silicon (and Xilinx offers much bigger devices). They all can be configured during the same configuration cycle and then clocked with the same or different clocks. But clearly they are all on, they are all available and they may be connected (paralleled). Xilinx provides plenty of interconnect, so the challenge becomes one of the designer's ingenuity.

A HUMBLE approach constructs a computing engine that exploits parallelism upon a foundation that is fundamentally parallel. So let's look at three parts of HUMBLE that run in parallel.

The first is a portion of HUMBLE's dual hot-pluggable and hot-swappable PS2 interface. Within this part is a CORE Generator™ FIFO for temporary storage of received values. There was no need to design the FIFO from scratch, because Xilinx offers a parameterizable core that will work with less effort. After starting COREGen and selecting Memories & Storage Elements – FIFOs (Fifo Generator), I entered the component name and desired parameters. Then I pressed Finish, and COREGen delivered a ready-to-use FIFO that synchronously and elastically stores the number of values desired.

Using this same mind-set, I've created a development system and opened access to the HUMBLE intellectual property. The HUMBLE PC Development System adds to the Xilinx cores by including a basic set of components written by Electronic Compute Systems, Inc. as well as library access to HUMBLE's more-sophisticated cores. Thus, we've made the building blocks of HUMBLE accessible from a library of basic components. Our Web site, *www.ecs-pc.com*, offers a list of library components and cores (look under Services & Cores).

For design entry when using cores, all we need is a component declaration and a link to its library (for the compiler). Here is the VHDL that describes the instantiation for this Xilinx CoreGen FIFO within the hot-pluggable and hot-swappable HUMBLE PS2 interface:

```
U40: fifo8x256

PORT MAP(

  Clk          => Clk100,

  Sinit        => Reset,

  Din          => RxData(8 DOWNTO 1),

  Wr_en        => DevWrite,     --
  Rd_en        => PushBDBreg(2),

  Dout         => DevFifoout,

  Full         => DevFifoFull,

  Empty        => DevFifoEmpty,

  Data_Count   => DevFifoCount

  );
```

Consider that two of these buffers are running in parallel. One is used for the key-

board and the other for the mouse. Hence, they are running in parallel with respect to each other and everything else, and are both "on" immediately after configuration.

A second example is the pipelined NoBL (no bus latency), also called zero bus turnaround (ZBT), static-RAM interface. It uses a Cypress CY1354B SRAM. This interface worked the first time without modification. So let's look at HUMBLE's control signals for this NoBL SRAM.

The Xilinx ML401 board shares the data bus between linear flash and NoBL SRAM, so to use both requires care. This sharing is a design trade-off of the ML401 board, not of HUMBLE. As it turns out, it's a reasonable compromise on Xilinx's part, given all of the other pin- and functional-related requirements vs. the number of pins available. Here is the VHDL for the NoBL SRAM control signals that synthesize to logic cells and slices:

```
CEtmp   <= '1' WHEN OptionsReg(0) =
SRAM AND (ReadSRAM = '1' OR
WriteSRAM = '1') ELSE '0';
-- Clock Enable

Bwatmp  <= '1' WHEN WriteSRAM = '1'
ELSE '0';  -- byte write selects

Bwbtmp  <= '1' WHEN WriteSRAM = '1'
ELSE '0';

BWctmp  <= '1' WHEN WriteSRAM = '1'
ELSE '0';

Bwdtmp  <= '1' WHEN WriteSRAM = '1'
ELSE '0';

Wetmp   <= '1' WHEN WriteSRAM = '1'
ELSE '0';  -- wr enable

Oetmp   <= '1' WHEN ReadSRAM  = '1'
ELSE '0';  -- output enable

Ce1tmp  <= '1' WHEN OptionsReg(0) =
SRAM AND (ReadSRAM = '1' OR
WriteSRAM = '1') ELSE '0';
-- Chip Enable

Ce2tmp  <= '1' WHEN ReadSRAM = '1'
OR WriteSRAM = '1' ELSE '0';

Ce3tmp  <= '1' WHEN ReadSRAM = '1'
OR WriteSRAM = '1' ELSE '0';

CE      <= CEtmp ;

Bwa     <= Bwatmp;

Bwb     <= Bwbtmp;

BWc     <= BWctmp;

Bwd     <= Bwdtmp;

We      <= Wetmp ;

Oe      <= Oetmp ;

Ce1     <= Ce1tmp;

Ce2     <= Ce2tmp;

Ce3     <= Ce3tmp;
```

This is a synchronous interface to and from a synchronous SRAM. All of the signals above are coming from or going to registers. Packing the IOB registers is assumed. Corresponding polarity flips as per the device datasheet are also assumed. Notice that the OptionsReg(0) signal makes the NoBL SRAM bus cycles "chip-enable controlled," steering them toward the SRAM or, conversely, to linear flash. Also notice that there is no need for byte-wide write enables thus far, so they actually have a common driving-signal description. While it doesn't need to be as verbose as shown, we can rely on synthesis to simplify it, or come back at some later time and manually do the same. However, if it should make sense in the future, byte-wide control is possible and I have retained the hooks for it. The SRAM control signaling is fairly simple and resides in the logic cells and slices.

As the third example within the TCP/IP portion of HUMBLE known as ipMoE (Internet Protocol Over Ethernet), there is a need to complement a checksum value. Here is the line of VHDL used within ipMoE that synthesizes to logic cells and slices.

```
U263:OnesCompl16 PORT MAP(CheckSum,
CheckSum1sCompl);
```

And here is the corresponding VHDL code for performing the one's complement.

```
LIBRARY ieee;
USE ieee.Std_Logic_1164.ALL;
USE ieee.Std_Logic_unsigned.ALL;
ENTITY OnesCompl16 IS
PORT(
        Ain     :IN
Std_Logic_Vector(15 DOWNTO 0);
        Cout    :OUT
Std_Logic_Vector(15 DOWNTO 0)
        );
END ENTITY OnesCompl16;

ARCHITECTURE ArchOnesCompl16 OF
OnesCompl16 IS
-- Internal nodes
SIGNAL Aintmp  :Std_Logic_Vector(15
DOWNTO 0);
BEGIN
-- The following describes comple-
menting each bit
g0:FOR i in 0 to 15 GENERATE
        Aintmp(i) <= NOT Ain(i);
END GENERATE g0;
Cout <= Aintmp;
END ArchOnesCompl16;
```

These three examples span four interfaces, including mouse, keyboard, NoBL SRAM and Ethernet. They all operate in parallel, which makes sense for overall operation, because they reside individually in Xilinx logic cells, slices or cores, which themselves are fundamentally all simultaneously on. Hence, using Xilinx logic cells with a set of HUMBLE basic components and more-sophisticated cores makes it possible to exploit parallelism, unleashing things previously thought too difficult or impossible in a PC.

Certainly these examples do not reveal all the inner workings of the embedded HUMBLE PC. This is why we offer library access to these basic components and more-sophisticated cores. I hope these examples together with my explanations will begin to give a sense of the innovation and its potential.

We are actively seeking embedded and PC partners with an eye toward making the HUMBLE PC full-featured. If you are interested in knowing more, consider signing a nondisclosure agreement with us. Also, feel free to contact us about your embedded application. To see a demonstration of the HUMBLE PC, visit *www.ecs-pc.com*. To view the HUMBLE PC in action, see *www.youtube.com/watch?v=i_v6IaD6-Ts*.

# Interpolated Lookup Tables: Simple Way to Implement a DSP Function

## If a digital signal processor core doesn't have precisely the functionality you need, use an ILUT to bridge the gap.

by Daniele Bagni
DSP Specialist FAE
Xilinx, Inc.
*daniele.bagni@xilinx.com*

As a Xilinx FAE, I'm often asked if we offer a DSP core that has functionality particularly suited for all of a customer's design requirements. Sometimes a core may be too large, too small or not fast enough. Sometimes, we'll have a core in development that will exactly fit their needs and soon be available in CORE Generator™. But even in that scenario, customers usually want a particular set of DSP functionality now, and don't want to wait. In these cases I often suggest they use the interpolated lookup tables in our devices to customize their DSP functionality.

A lookup table (LUT) is a memory element that "looks up" what the outputs should be for any given combination of input states, to ensure there are definite outputs for every input. Using LUTs to perform DSP functions offers some major advantages:

- You can change the LUT content in a high-abstraction-level programming language such as MATLAB® or Simulink®.
- You can design a DSP function to perform math functions that would be extremely difficult to emulate with discrete logic operations, for example: y=log(x), y=exp(x), y=1/x, y=sin(x) and so on.
- LUTs are also a simple way to implement a complicated mathematical function that could require too many FPGA resources in terms of configurable logic block (CLB) slices and embedded multiply or DSP48 programmable multiply-and-accumulate (MAC) units.

But of course, using LUTs in this manner has some disadvantages, too. When you implement DSP functionality in LUTs you have to use block-RAM (BRAM) elements. Implementing the y=sqrt(x) function for 16-bit input and 18-bit output variables, respectively x and y, requires roughly 64 BRAM units of 18 Kbits each. Giving up that amount of BRAM may be too costly from a system architecture point of view if, for example, you are targeting a small Spartan® device or if you have too many operations to implement and can't spare 64 BRAM units for each one.

The interpolated LUT approach delivers the advantages of a LUT-based function implementation without using too much BRAM. In this technique, you use contiguous outputs from a smaller LUT—for example, one that is 1K words deep—and linearly interpolate it to emulate a larger LUT. In doing so, you can achieve a finer numerical resolution than is possible with a 1K-word LUT. This makes the cost of using a LUT much more reasonable, as the implementation only takes one BRAM and one embedded multiplier (or DSP48) plus a few CLB slices for the control logic. In addition, the numerical accuracy will be more than satisfactory from the point of view of signal-to-noise ratio (SNR).

Of course, applying the interpolated LUT (ILUT) technique takes a bit of know-how. As an example, using the approach on the y=sqrt(x) function will show clearly how the ILUT performs in terms of area occupation, timing and numerical accuracy. After walking through that exercise, I will review some practical examples where I have applied this approach for customers with completely different demands—specifically, in the linearization of a sensor with a nonlinear transfer function and in an implementation of an adaptive finite impulse response (FIR) filter to remove speckle noise from synthetic aperture radar (SAR) images.

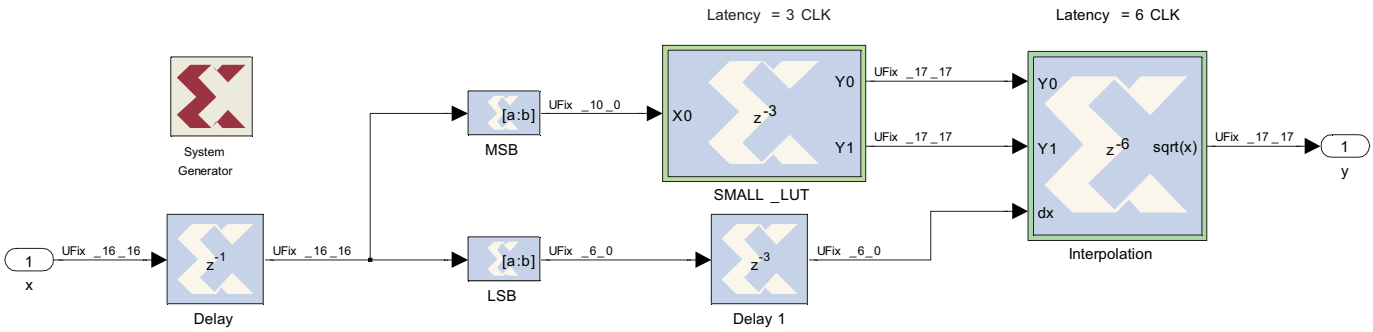



*Figure 1 – Top-level scheme of interpolated lookup table in System Generator for DSP*

## Designing in System Generator for DSP

To implement a DSP algorithm on a Xilinx FPGA, I use the System Generator for DSP design and synthesis tool, which works within the Simulink model-based methodology from The MathWorks. System Generator benefits from the Xilinx DSP blockset for Simulink and will automatically invoke CORE Generator to produce highly optimized netlists for the DSP building blocks. Simulink is a double-precision floating-point design tool, whereas System Generator is a fixed-point arithmetic tool. Nevertheless, using these tools together, you can define the total number of bits and the binary position of every signal, and can therefore manipulate fractional numbers in fixed-point arithmetic. The simulation results are cycle-accurate and bit-true, so you can easily compare them to floating-point reference results generated with either MATLAB scripts or Simulink blocks to check for quantization errors.

Figure 1 shows the System Generator top level of the ILUT scheme. To make the approach as general as possible, the input variable x of nx=16 bits is supposed to be normalized between 0 (included) and 1 (excluded); therefore, its format is "unsigned 16 bits in total with 16 bits to the right of the binary point," and is called the Ufix_16_16 format. The most significant bits (MSB) and least significant bits (LSB) blocks respectively extract nb=10 MSB and nx-nb=6 LSB from the input data. These signals are called x0 and dx. The output y=sqrt(x) is represented with ny=17 bits, normalized into the Ufix_17_17 format.

Figure 2 illustrates the stage with the small LUT with a depth of 1K words that I've implemented via a dual-port RAM block. Since such a block is applied as a read-only memory (ROM), the Boolean constant block, We_const, forces the write enables to zero. The signals $x_0$ and $x_0$+1 are used as two subsequent addresses to the ROM table. The zero constant of the Data_const block defines the size of any ROM word (ny, in our case).

The following formula illustrates how to linearly interpolate a point at coordinates (x, y) between two known points ($x_0$, $y_0$) and ($x_1$, $y_1$), with x0 the MSB of x:

$$y = y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)\cdot(x - x_0) = y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right)\cdot(y_1 - y_0)$$

$$= y_0 + \frac{x - x_0}{2^{-nb}}\cdot(y_1 - y_0)$$

Note that $x_1$ and $x_0$ are two contiguous addresses to the small LUT and they are one LSB apart from each other. Since the small LUT has nb bits of addressing space, such LSB has the value $2^{-nb}$.

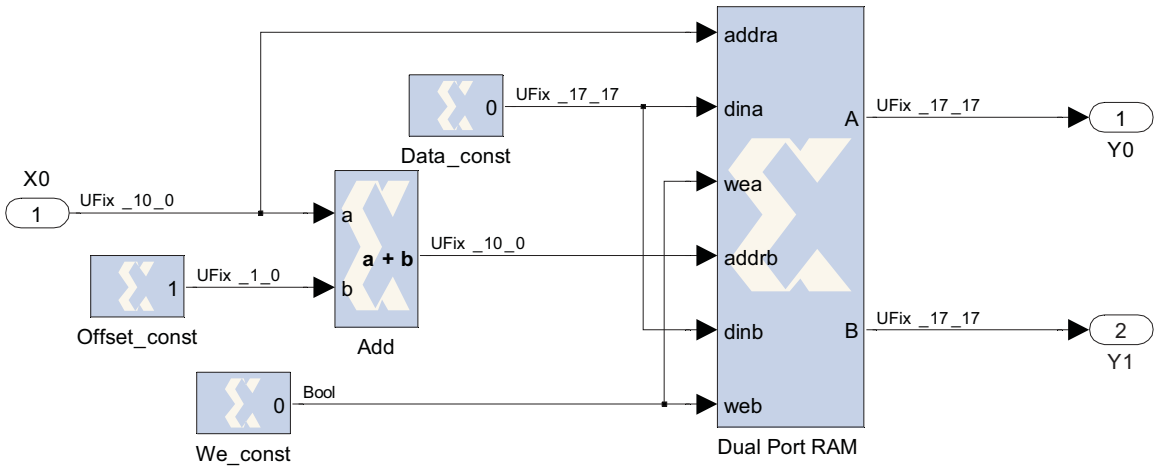



*Figure 2 – Scheme of the smaller LUT in System Generator for DSP*

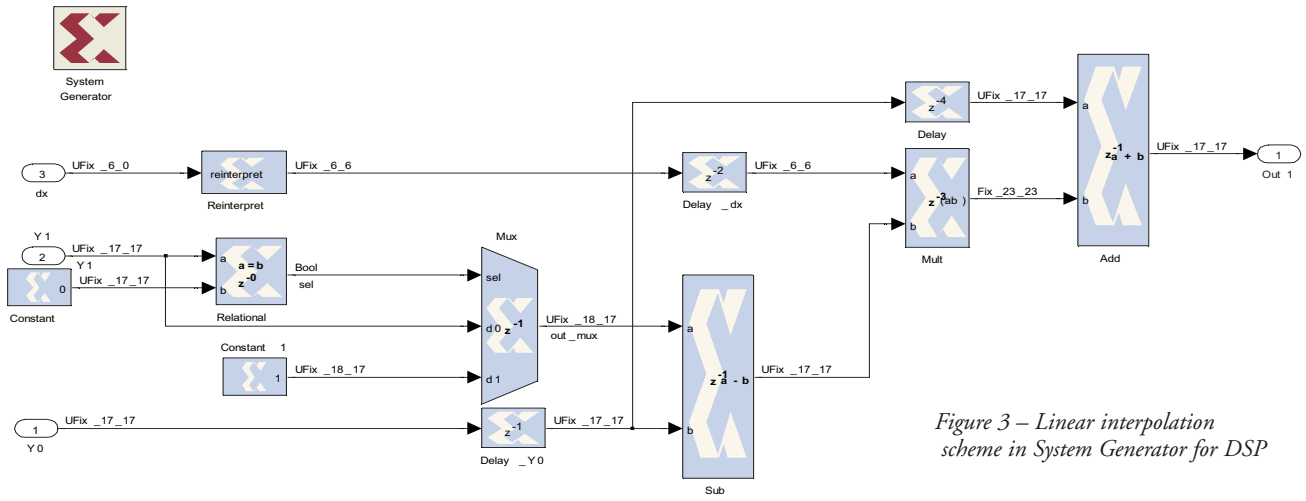



*Figure 3 – Linear interpolation scheme in System Generator for DSP*

The interpolation stage is shown in Figure 3. The Reinterpret block changes the dx=x-x0 signal without altering the binary representation; it relocates the binary point (from UFix_6_0 to UFix_6_6 format) and outputs a fractionary number of nx-nb bits, thus computing the value (x-x0)/ $2^{-nb}$.

In hardware, this block costs nothing. Generally speaking (and depending on what function we're applying using an ILUT approach), if $y_1$=0 and $y_0$=0, we force $y_1$-$y_0$=1 so that we get $1/2^{-nb}$ instead of zero. We perform this action with the Mux, Relational, Constant and Constant1 blocks. The remaining Mult, Add and Sub blocks execute the linear interpolation formula. In this case, I forced the output signal from the Mult block to have 17-bits resolution and not 23 as theoretically required, because the overall numerical accuracy is good enough for the experiment. Furthermore, all the results are unsigned, since the y=sqrt(x) function is monotonically increasing. In other words, different functions may require different tuning of the data types, but won't deviate dramatically from the principles of the scheme in Figure 3.

Assuming the Spartan-3E 1200 (fg320-4) is our target device and we're using the ISE® Design Suite and System Generator for DSP 10.1SP3 release tools, the FPGA resources occupied after place and route are summarized here:

```
Design Summary using target part "3s1200efg320-4"
Logic utilization:
   Number of slice flip-flops:   198 out of 17344 1%
   Number of four-input LUTs:    086 out of 17344 1%
Logic distribution:
   Number of occupied slices:    111 out of 08672 1%
   Number of MULT18X18:          001 out of 00028 3%
   Number of BRAMs:              001 out of 00028 3%
```

The design is fully pipelined and can deliver a new output result at any clock cycle. The latency is 10 clock cycles and the maximum data rate achievable is 194.70 MSPS (millions of samples per second). In terms of numerical accuracy, the ratio between the power of reference floating-point results and the quantization error of the System Generator for DSP fixed-point output yields an SNR of 71.94 dB or 77.95 dB, for an ILUT with a depth size of 1K or 2K words respectively.

Alternatively to the ILUT, we can apply the CORDIC SQRT Block from the Xilinx Reference Math Blockset in System Generator for DSP. In this case the overall latency is 37 clock cycles, max data rate is 115.18 MSPS and area resource occupation is 940 slice flip-flops. There are 885 four-input LUTs, 560 occupied slices and two MULT18x18 embedded multipliers. The SNR is 40.64 dB. These results show that CORDIC is a fantastic way to implement math operations in fixed point. But the ILUT approach is even better in many cases.

## Linearizing Nonlinear Sensors

Nowadays a lot of companies use "smart sensors" in industrial control systems requiring low area, low power consumption and high performance, together with the lowest possible cost and reduced development time. A generic smart sensor can be seen as one functional unit composed of a sensor and its signal-conditioning circuitry, an analog-to-digital converter (ADC) and an associated DSP subsystem (with or without an additional embedded processor)—all integrated in the same device, as shown in Figure 4.

The smart sensor has to convert physical quantities—for example, currents in an electric motor—into digital signals that digital electronic circuits can process. Certain features of the components and the technology used to construct these sensors commonly cause errors such as offset, gain and nonlinearity. Such errors produce an overall transfer function that is often not linear.

Frequently, customers correct these errors within the DSP subsystem running in their products. If y=f(x) is the digital output signal from the sensor and ADC cascade, the DSP subsystem has to compensate for a nonlinear function by performing its inverse $g(y)=f^{-1}(y)$, so that the overall output z becomes

$$z = m \cdot g(y) + b = m \cdot f^{-1}(y) + b = m \cdot f^{-1}\left(f(x)\right) + b = m \cdot x + b$$

which is the equation of a straight line with slope m and vertical intercept b.

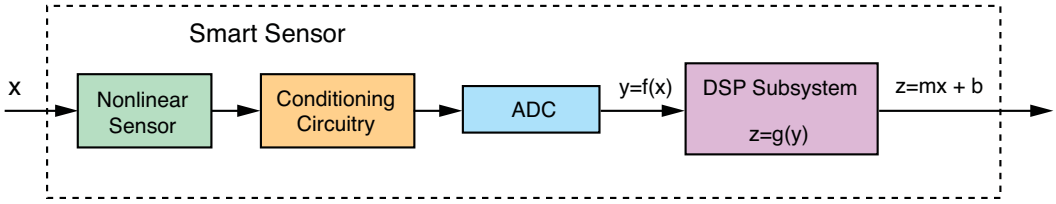



*Figure 4 – Block diagram of a smart sensor*

The simplest linearization method is the LUT approach, with the sensor calibration points stored in a ROM. However, for a 16-bit ADC, the ROM size would be too large, requiring 64 BRAM units. This is where an interpolated LUT is a good solution.

Let's assume, as an example, that the nonlinear transfer function is a parabola. The next MATLAB fragment code explains how to generate the m and b parameters of the final straight line and how to compute g(y), the inverse function of f(x). Figure 5 illustrates the three curves in different colors. Note that the inversion process of f(x) can cause missing values in g(y). That's because there can be several points with the same y value corresponding to distinct, different x points. Therefore, g(y) has to be smoothed to fill all the possible holes. (For reasons of conciseness, I haven't included this part of the operation in the MATLAB fragment.)

```
ny = 18;  % 18 bits per word of the ILUT applied for sensor linearization
nx = 16;  % number of bits of the sensor ADC output codes
% emulation of sensor nonlinear transfer function
x = -1.5 : 3 / 2^nx : 1.5 -3/2^nx;
y = -0.196 + x + 0.1 * (x.^2);
% parabola y=f(x)
min_x = min(x); max_x = max(x); min_y = min(y); max_y = max(y);
% histogram stretching of y curve to use all the 2^N available ADC bits
yy      = ((y-min_y) .* (2^nx-1))/(max_y-min_y);
yq      = round(yy);  % to have a purely integer transfer function

figure; plot(x, yq, 'k', 'LineWidth',2); title 'parabola y=f(x)';

% linear regression of y = f(x) to determine the slope and y-intercept of
% the equation y = m*x + b of a straight line; at the end we will get:
% m = 2.184494117400677e+004 and b = 2.952424998204837e+004
% with a correlation factor r = 0.997036366735360 (values close to 1 indicate excellent
% reliability of the linear regression)
%
p = polyfit(x, yq, 1); m = p(1); b = p(2);

% let us plot the line approximating the y curve: z = b + m * x;
z2 = b + m .* x;       % generic straight line defined by b and m
zq = z2 + abs(z2(1)); % vertically translated to have same intercept of parabola f(x)
zq = round(zq);
hold on; plot(x, zq, 'g', 'LineWidth',2); grid;
title 'parabola y=f(x) and line y=m*x+b'; hold off;

x_of_y = -1 * ones(2^nx, 1); % memory allocation for g(y)
addr   = uint32(yq(1:end));

% this is g(y) reverse function of the non linear sensor curve y=f(x)
x_of_y(addr(2:end)) = ( (x(2:end)-min_x).*2^ny)/(max_x-min_x);
x_of_y(1)=0; % addr(1)=0 is forbidden in MATLAB

% x_of_y has to be smoothed before being used (not shown here)
figure; plot(yq, (x_of_y), 'b', 'LineWidth',2); title 'original x=g(y)'; grid

% reference LUT with values to linearize the sensor non linear transfer function
refLUT = round( m .* (min_x+ smoothed_x_of_y*(max_x-min_x)/2^ny) +b +abs(z2(1)) );
```

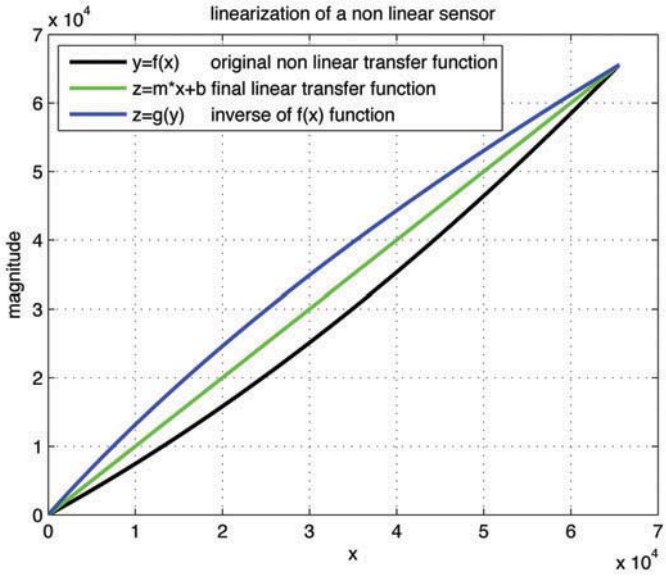



*Figure 5 – The parabola emulating the nonlinear sensor transfer function f(x) is shown in black. The straight green line represents the final sensor obtained by the linearization DSP subsystem, which applies the inverse g(y) function shown in blue.*

Running the fixed-point cycle-based simulation in System Generator for DSP, I obtain a 92.48-dB SNR on the whole output range of the nonlinear sensor, with a design pretty similar to the one reported in Figures 1-3.

*Figure 6 – Speckle noise affects the image on the left; on the right side is the filtered image.*

### Speckle Noise Reduction

Tracking a target object from a high-speed moving system—namely, a missile—is a challenging task requiring a highly sophisticated DSP algorithm and different types of acquisition media, such as synthetic aperture radar sensors. As is typical of all electromagnetic coherent sources (such as lasers), SAR imaging devices suffer from speckle noise. Therefore, the first stage of any SAR-based DSP chain is a bidimensional (2D) adaptive FIR filter to reduce that noise (it isn't possible to completely remove it). Figure 6 shows a MATLAB simulation of speckle noise. The noise synthetically deteriorated the image on the left; the right image is the output of the 2D FIR filter golden model.

Speckle noise is a multiplicative noise following an exponential distribution and is completely defined by its variance σ. Hence, a widely adopted anti-speckle noise method is the Frost filter, named for author V. S. Frost, who wrote a paper on the phenomenon in 1981. In the case of a 3x3 block size, it can be modeled by the formula:

$$y_{ij} = \sum_{ij} x_{ij} \cdot \exp\left( -K \cdot \frac{\sigma^2}{\mu_1^2} \cdot T_{ij} \right)$$

$$with \quad T_{ij} = \begin{bmatrix} \sqrt{2} & 1 & \sqrt{2} \\ 1 & 0 & 1 \\ \sqrt{2} & 1 & \sqrt{2} \end{bmatrix}$$

with $x_{ij}$ and $y_{ij}$ representing the input and output samples, respectively, of the Frost filter. K is a gain factor to control the filter strength (for simplicity's sake, I assume in the following that K=1), $\mu_1$ and σ are respectively the mean and variance values of the 2D kernel, Tij is the matrix of the distance from the central output pixel (of index ij=22) and all the surrounding ones. The next equations illustrate that the key factor in implementing such a filter is R1, the ratio between the momentum of first- and second-order $\mu_1$ and $\mu_2$ in the 3x3 block:

$$\mu_1 = \frac{1}{N}\sum_i x_i \quad \mu_2 = \frac{1}{N}\sum_i x_i^2$$

$$\sigma^2 / \mu_1^2 = \frac{\mu_2}{\mu_1^2} - 1 = N \cdot \frac{\sum_i x_i^2}{\left(\sum_i x_i\right)^2} - 1 = N \cdot R_1 - 1 = R$$

$$with \quad 0 \le R_1 < 1 \quad and \quad 0 \le R < 8 \quad for \quad N = 9$$

R1 has a range between 0 and 1, and I have found experimentally that it can be represented with 16 to 20 bits to get good numerical accuracy.

Once I designed the stage to compute R1 in System Generator for DSP, I decided to implement the normalized filter coefficients via interpolated LUTs. The content of the LUTs is shown in the following MATLAB code:

```
nb = 10; % number of bits to address the LUT

% input addresses to the LUT
R1 = 0 : 1/(2^nb) : 1-1/(2^nb);

R = 9*R1 - 1;           % R = N * R1 -1 with N=9
ind_R = (R<0); R(ind_R)=0; % just to be sure that R >= 0
always

x = R*sqrt(2);

M11_LUT = exp( -x );  % coeff. for diagonal indexes
ij=11, 13, 31, 33
M12_LUT = exp( -R );  % coeff. for vert. & horiz. Indexes
ij=12, 21, 23, 32

tot = 4*(M11_LUT + M12_LUT) +1; % sum of all coeff.,
including index ij=22

norm_M11_LUT = M11_LUT ./tot; % normalized coeff. for diag-
onal (ij=11, 13, 31, 33)
norm_M12_LUT = M12_LUT ./tot; % normalized coeff. for vert.
& horiz. (ij=12, 21, 23, 32)
M22_LUT      = 1 ./ tot;    % normalized coeff. for cen-
tral position (ij=22)

figure; plot(x, norm_M11_LUT, 'r', x, norm_M12_LUT, 'g', x,
M22_LUT, 'b');
grid; title 'normalized M11 (red), M12 (green), M22 (blue)'
```

Figure 7 illustrates the curves of the normalized coefficients along the R1 input signal. There are only three curves, since the Tij matrix is symmetric around the central pixel of index ij=22. The numerical results exhibit SNR values from 81.28 to 83.38 dB, depending on the curve, when compared against purely floating-point reference models. For the curious reader, the following MATLAB fragment code shows the 2D filter processing (for conciseness, ILUT functions are not included).

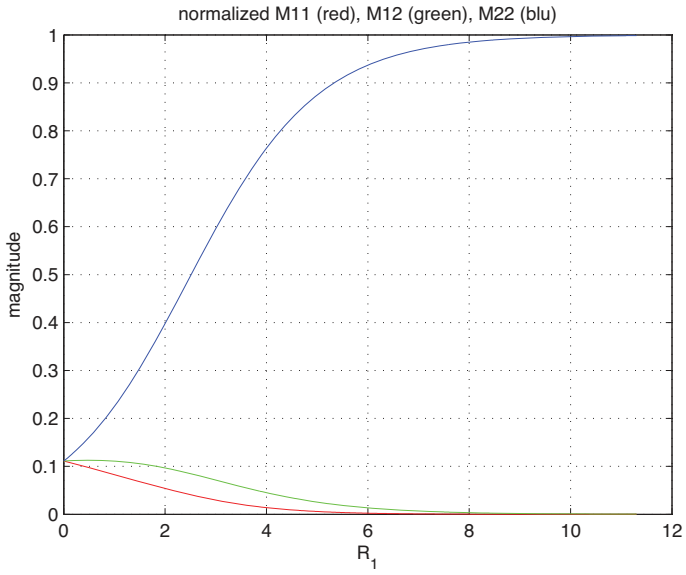



*Figure 7 – Normalized coefficients along R1 parameter of the speckle noise reduction filter*

```
function [out] = synth_frost3x3(inp) % inp is a vector of 9 samples

mu1=0; mu2=0;
for k = 1 : 9
     mu1  = mu1 + inp(k);     % first  order momentum (or mean value)
     mu2  = mu2 + inp(k)^2;   % second order momentum
end

R1 = mu2 / (mu1^2 +1); if (tmp_R1<0) R1=0; end % since R1>=0 always

% in MATLAB array indexing goes from 1 to 1024 (not from 0 to 1023 as in HW)
if (R1 >=1024) addr=1024; elseif (R1==0) addr=1; else addr=R1+1; end

% the ILUT functions are not shown here
M11 = M11_ILUT(addr);  M12 = M12_ILUT(addr); M22 = M22_ILUT(addr);

reg11 = inp(1) + inp(3) + inp(7) + inp(9); % pre-add pixels of index 11, 13,
31, 33
reg12 = inp(2) + inp(4) + inp(6) + inp(8); % pre-add pixels of index 12, 21,
23, 32
out11 = M11 * reg11;        % sum of filtered pixels index 11, 13, 31, 33
out12 = M12 * reg12;        % sum of filtered pixels index 12, 21, 23, 32
out22 = M22 * inp(5);       % central pixel of index 22 in the 3x3 block
out   = out11 + out12 + out22; % filter output pixel
```

In short, these examples show that interpolated lookup tables are a simple yet powerful method for implementing DSP functions in Xilinx FPGAs. They can help you achieve very good numerical accuracy (SNR) and high data rates while keeping area occupation relatively low.

***Acknowledgements***

*I would like to thank my colleague Michel Pecot, who first introduced me to the interpolated lookup table approach with his Gamma Correction design in System Generator for DSP, when I joined Xilinx three years ago.*

*Daniele Bagni is a DSP Specialist FAE in the Xilinx Milan sales office in Italy. What Daniele enjoys most about his job is providing customers with feasibility studies and facing a large variety of DSP applications and problems. In his spare time, he likes to play tennis. At home he loves biking or playing table tennis with his wife and two sons.*

# Sign Up for X-fest 2009

## Seminars in 36 cities center on Virtex-6, Spartan-6 and Xilinx Targeted Design Platforms.

If you want to get up to speed fast on Xilinx® Targeted Design Platforms, don't miss X-fest 2009, a 36-city global technical seminar series offering practical, how-to training for FPGA, DSP and embedded-systems designers. The events will provide information on how to design with the Xilinx Virtex®-6 and Spartan®-6 FPGA families, ISE® Design Suite software, Xilinx and third-party IP, evaluation kits and reference designs.

The Avnet Electronics Marketing operating group and Xilinx have now opened registration for X-fest 2009. Seminars will take place in locations around the globe beginning in October and ending in February 2010. The events scheduled for North America, Europe and Israel are listed in the table.

This series of free one-day seminars will feature multitrack training sessions ranging from building-block to system-level solutions that use the latest FPGA technologies. Attendees will discover the advantages of the new Xilinx Targeted Design Platforms and their constituent parts, including cutting-edge FPGA devices, IP and design tools.

You will have access to world-class industry experts and partner exhibits from Cypress Semiconductor, Intel, Maxim Integrated Products, National Semiconductor, NXP, Texas Instruments and Tyco Electronics. Designers will leave X-fest armed with the tools and practical knowledge you need to tackle existing and future design challenges, and extend your product's life cycle.

X-fest 2009 will include several interesting courses. In "Interfacing DDR3 and LPDRAM with the Xilinx Spartan-6 Hard Memory Controller," you will learn about the built-in memory controller in the Spartan-6, its features and how to connect logic to the hard core to achieve maximum performance with DDR3. Also, the class will describe how to implement the low-power DDR memory for power-saving applications.

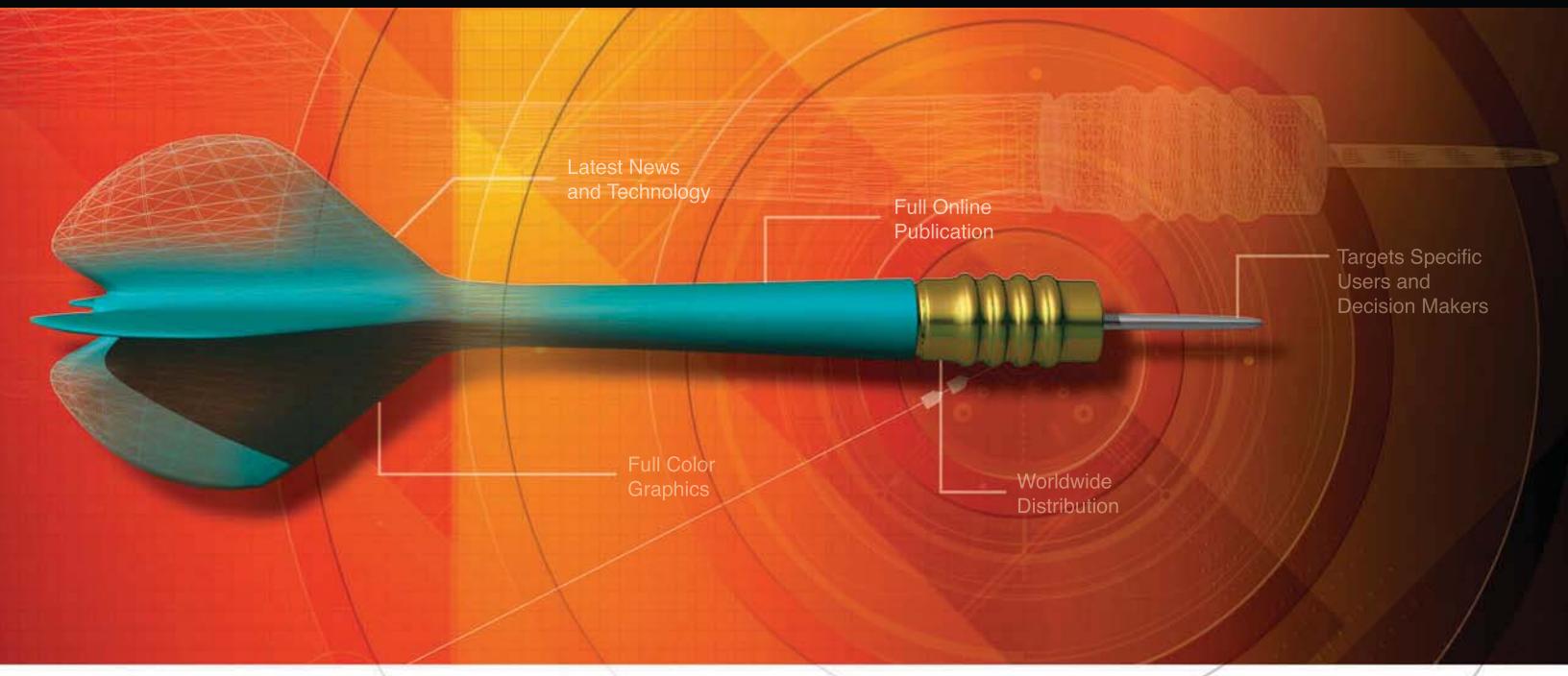
| | City | Delivery Date |
|---|---|---|
| **Americas** | Austin, Texas | Tuesday, Nov. 3 |
| | Baltimore | Thursday, Oct. 8 |
| | Boston | Thursday, Oct. 22 |
| | Dallas | Tuesday, Oct. 20 |
| | Denver | Thursday, Nov. 5 |
| | Irvine, Calif. | Thursday, Oct. 29 |
| | Melbourne, Fla. | Tuesday, Oct. 20 |
| | Minneapolis | Tuesday, Oct. 13 |
| | San Diego | Wednesday, Oct. 28 |
| | San Jose, Calif. | Tuesday, Oct. 6 |
| | Toronto | Thursday, Oct. 15 |
| | Vancouver, B.C. | Wednesday, Nov. 18 |
| **Europe, Israel** | Antwerp, Belgium | Thursday, Nov. 26 |
| | Århus, Denmark | Tuesday, Dec. 1 |
| | Leipzig, Germany | Tuesday, Nov. 24 |
| | Madrid, Spain | Thursday, Nov. 19 |
| | Milan, Italy | Thursday, Nov. 12 |
| | Munich, Germany | Wednesday, Nov. 11 |
| | Oslo, Norway | Thursday, Nov. 5 |
| | Paris | Wednesday, Nov. 18 |
| | Silverstone, U.K. | Tuesday, Nov. 3 |
| | Tel Aviv, Israel | Monday, Nov. 9 |
| | Warsaw, Poland | Tuesday, Nov. 17 |

In another course, "High-speed Clocking: Challenges, Pitfalls and Solutions," you'll learn about the many on-chip resources of the Spartan-6 and Virtex-6 FPGAs, including the mixed-mode clock managers, phase-locked loops, digital clock managers, BUFH, GCLK and BUFG. You will learn when you should use each of them and why. The class will also review several external clocking solutions, why they are necessary and how they support high-performance sampling systems.

Yet another course, "Xilinx Networking: 10/100/1000 to Real-Time," aims to help designers tackling network connectivity in the embedded realm. Many designers in this space struggle to satisfy operational and performance goals within their cost, power and real estate budgets. In addition, many networking designs have unique demands that affect the construction of both hardware and software platforms. For example, applications such as EtherCAT or SerCOS-III require specialized IP or the use of IEEE-1588 components for real-time network transactions. This course will examine multiple networking options, starting from the lowest hardware layers to the top of the TCP/IP protocol stack. It will also discuss the advantages of including an FPGA in your system design.

For complete information about X-Fest 2009 including full course descriptions, locations and registration information, please visit *http://www.weboom.com/avnet/index.html.*

# Get on Target




## Let Xcell Publications help you get your message out to thousands of programmable logic users.

Hit your marketing target by advertising your product or service in the Xilinx *Xcell Journal*, you'll reach thousands of engineers, designers, and engineering managers worldwide!

The Xilinx *Xcell Journal* is an award-winning publication, dedicated specifically to helping programmable logic users – and it works.

We offer affordable advertising rates and a variety of advertisement sizes to meet any budget!

Call today :
(800) 493-5551
or e-mail us at
*xcelladsales@aol.com*

Join the other leaders in our industry and advertise in the *Xcell Journal*!

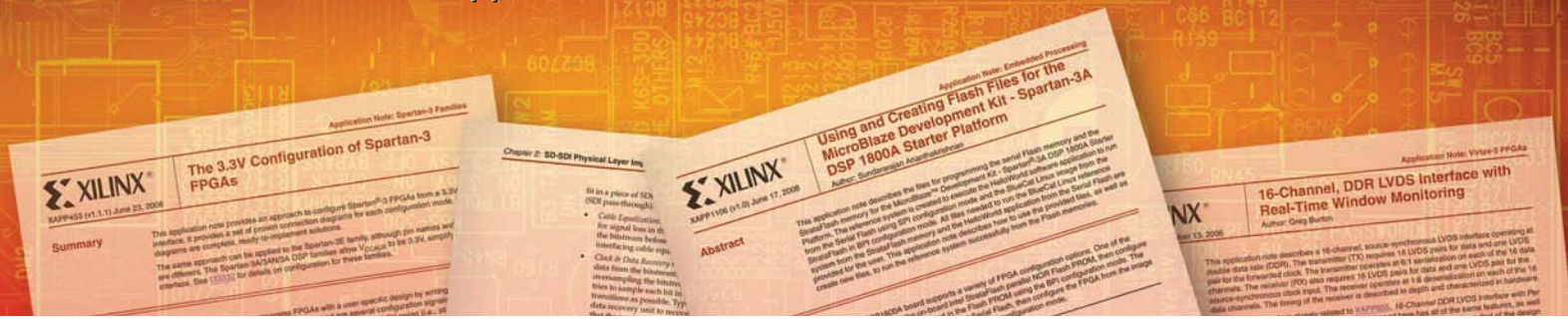

Xcell Publications

**www.xilinx.com/xcell/**

XILINX®

# Application Notes

## If you want to do a bit more reading about how our FPGAs lend themselves to a broad number of applications, we recommend these notes.




### XAPP1020: Post-Configuration Access to SPI Flash Memory with Virtex-5 FPGAs

*http://www.xilinx.com/support/documentation/application_notes/xapp1020.pdf*

Virtex®-5 FPGAs support direct configuration from industry-standard Serial Peripheral Interface (SPI) flash memories. After configuration, it is possible for your application to read and write to this memory for general-purpose use. However, before the application can communicate with the SPI flash memory, it must first instantiate the STARTUP_VIRTEX5 primitive to gain access to some of the signals connected to the memory. In this application note, Daniel Cherry uses a reference design to explain the techniques you can use to implement the STARTUP_VIRTEX5 primitive and interface it with an external SPI flash memory. In particular, this application note walks you through the instantiation process for implementing the STARTUP_VIRTEX5 primitive and a software application to demonstrate how, after you configure it, you can use the SPI flash memory.

The reference design targets the ML505 evaluation board, which includes a 32-Mbit Numonyx (formerly STMicroelectronics) M25P32 serial flash memory. For custom applications, we recommend you use an SPI flash memory that is supported by the iMPACT configuration software and then confirm device functionality with iMPACT before testing this reference design.

### XAPP1107: Getting Started Using Git

*http://www.xilinx.com/support/documentation/application_notes/xapp1107.pdf*

Xilinx provides many offerings enabling customers to use Linux with Xilinx processor architectures. In addition to obtaining kernel sources from supported Xilinx third-party partners, you can also download kernel sources from the Xililnx Linux Git Tree, a distributed version control system commonly used for distributing Linux kernel sources.

In this application note, Kris Chaplin describes a way to set up a user environment for building Linux kernels using the Xilinx Git tree. To effectively use this application note, you must be running a Linux operating system that is compatible with the Xilinx ISE® Design Suite and EDK tools. The processes in this documentation are based on running Red Hat Linux 4 with the EDK and the ISE Design Suite 10.1 or newer. You must have the Xilinx EDK and ISE Design Suite for generating custom board hardware images.

### XAPP875: Dynamically Programmable DRU for High-Speed Serial I/O

*http://www.xilinx.com/support/documentation/application_notes/xapp875.pdf*

In this application note, authors Paolo Novellini and Giovanni Guasti show how to implement a noninteger data recovery unit (NI-DRU) in your Virtex-5 LXT, SXT, TXT and FXT FPGA-based designs to extend the data rate limit of RocketIO GTP and GTX transceivers in those FPGAs.

The NI-DRU extends the lower data rate limit to 0 Mbits/second and the upper limit to 1,250 Mbits/s, making embedded high-speed transceivers the ideal solution for true multirate serial interfaces. You can dynamically program the NI-DRU's operational settings (data rate, jitter bandwidth, input ppm range and jitter peaking), thus avoiding the need for bitstream reload or partial reconfiguration. Operating on a synchronous external reference clock, the NI-DRU supports fractional oversampling ratios. As such, you need only one BUFG, independent of the number of channels you are setting up, even if all channels are operating at different data rates.

Given the absence of a relationship between the reference clock and incoming data rate, two optional barrel shifters ease the inter-

facing of the NI-DRU with an external FIFO or with any required decoder. The first barrel shifter has a 10-bit output that you can easily couple to an 8B/10B or 4B/5B decoder (neither of which is included in the reference design). The second barrel shifter has a 16-bit output, and Xililnx specifically designed it for 8-bit protocols such as Sonet/SDH. You can also design your own barrel shifters.

The authors divide the application note into three main parts: the NI-DRU usage model, simulating the NI-DRU and testing the NI-DRU on the ML523 RocketIO™ transceiver characterization platform, revision C or higher. In the NI-DRU usage model section, the authors describe in detail a block diagram of the NI-DRU, in which they calculate the overall transfer function of the DRU as a function of all the hardware settings.

### XAPP1110: BFM Simulation of an EDK System Which Uses the PLBv46 Endpoint Bridge for PCI Express

*http://www.xilinx.com/support/documentation/application_notes/xapp1110.pdf*

In this application note, Lester Sanders and Mark Sasten demonstrate how to run a simulation using IBM CoreConnect Bus Functional Language (BFL) commands in an EDK system containing the PLBv46 Endpoint Bridge for PCI Express®. The simulation consists of a PCIe® downstream port model communicating over a PCIe link to an EDK system that contains the PLBv46 Endpoint Bridge for PCI Express. The bridge uses the Xilinx Block Plus Endpoint core for PCI Express in the Virtex-5 FPGA. A bus functional model (BFM) drives the EDK system.

Xilinx provides a simulation environment based on a downstream port model that has a test program interface. You can build this downstream port model with the Xilinx CORE Generator™ tool using prewritten programs and Verilog tasks to generate transaction-layer packets. In the note, Sanders and Sasten show how to set up the simulation and the steps you can use to run the system simulation with BFL commands. The note also provides example stimuli (specifically, root-complex-to-endpoint and endpoint-to-root-complex transactions) to test the PLBv46 Endpoint Bridge using the EDK system. The authors then show how to analyze the results from these tests in the waveform viewer.

This application note is a companion note to XAPP1111, "Simulation of an EDK System Which Uses the PLBv46 Endpoint Bridge for PCI Express."

### XAPP1111: Simulation of an EDK System Which Uses the PLBv46 Endpoint Bridge for PCI Express

*http://www.xilinx.com/support/documentation/application_notes/xapp1111.pdf*

In this companion note to XAPP1110, Lester Sanders shows how to run a simulation of an EDK system containing the PLBv46 Endpoint Bridge for the PCI Express core. Where application note XAPP1110 describes a bus functional model driving the simulation, this note shows how to use C code to do it. In particular, the simulation consists of a PCIe downstream port model communicating over a PCIe bus to an EDK system containing the PLBv46 Endpoint Bridge for PCI Express. You can build the downstream port model using the Xilinx CORE Generator tool. The PLBv46 Endpoint Bridge uses the Xilinx Block Plus Endpoint core for PCI Express in the Virtex-5 FPGA. C code running on the PowerPC® 440 drives the EDK system.

This application note is a companion note to XAPP1110, "BFM Simulation of an EDK System Which Uses the PLBv46 Endpoint Bridge for PCI Express."

### XAPP1014: Audio/Video Connectivity Solutions for Virtex-5 FPGAs: Reference Designs for the Broadcast Industry, Volume 2

*http://www.xilinx.com/support/documentation/application_notes/xapp1014.pdf*

This comprehensive, 558-page guide describes how you can use Virtex-5 FPGAs to implement various serial digital video interfaces commonly employed in the professional video broadcast industry. After several chapters introducing the various SMPTE standards, the guide delves into such topics as multirate SD/HD/3G-SDI using Virtex-5 FPGA RocketIO transceivers; SD-SDI using Virtex-5 FPGA SelectIO™ LVDS; DVB-ASI using Virtex-5 FPGA SelectIO LVDS, and AES Digital Audio. A final section discusses miscellaneous audio and video topics.

### XAPP1129: Integrating an EDK Custom Peripheral with a LocalLink Interface into Linux

*http://www.xilinx.com/support/documentation/application_notes/xapp1129.pdf*

In this application note, Brian Hill discusses using a LocalLink DMA peripheral with the Linux operating system, outlining the steps and methodology you will need to use a custom LocalLink scatter-gather DMA (SGDMA) core with Linux. The note provides a LocalLink loopback core with a Linux driver. Hill describes the driver design and operation at length.

### XAPP1130: Architecting ARINC 664, Part 7 (AFDX) Solutions

*http://www.xilinx.com/support/documentation/application_notes/xapp1130.pdf*

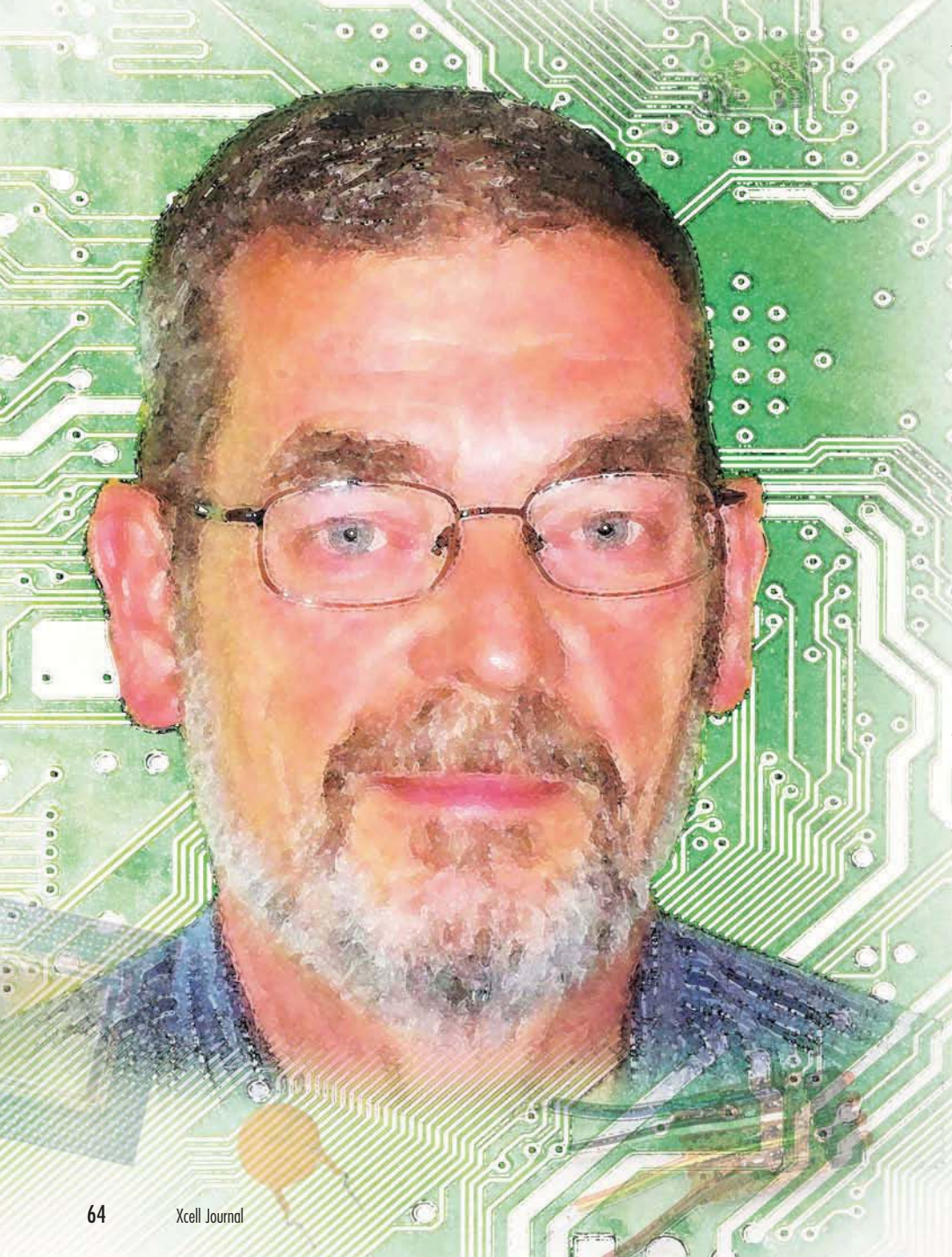
Each new generation of commercial aircraft has grown more complex, especially with the heavy reliance on fly-by-wire and the associated avionics. As more electronic systems are designed into airframes, traditional point-to-point wiring schemes are no longer practical. The designers of the Airbus A380 searched for a solution to reduce the amount of wiring, increase bandwidth and make use of commercial off-the-shelf (COTS) technology where possible. ARINC Specification 664, Part 7 is the result of that search.

In this application note, Ian Land and Jeff Elliott present a detailed overview of the architecture and function of avionics full-duplex switched Ethernet (AFDX), as defined in ARINC 664, Part 7. In addition, the authors provide a detailed description of how you can map various functional blocks required for an AFDX end system to both the Virtex®-4 and Virtex-5.

# Dick Corey Has the Engineering Bug

## Forget the golf clubs. Spartan-3A development board is diversion of choice for one newly retired EE.

by Mike Santarini
Publisher, *Xcell Journal*
Xilinx, Inc.
*mike.santarini@xilinx.com*



After spending his entire adult life working as an engineer, including stints in a number of small electronics companies in New York State and the last 24 years at Eastman Kodak Co., in Rochester, N.Y., Dick Corey retired three months ago, at the age of 62. He is already getting antsy, missing the trade, the lab, and is thinking of new design projects—and a return to the workforce.

At Kodak, Corey primarily worked on advanced research-and-development projects, with stints in product design. "Most of those were low-volume, large-equipment projects as opposed to consumer electronics, which Kodak is most noted for," said Corey. "I'm kind of a generalist. I do analog, digital and a bunch of software, usually to support my own hardware projects."

Corey comes from what he describes as "a nontraditional background" for someone who spent the majority of his career in advanced R&D. "I have an associate's degree from Rochester Institute of Technology," he said. "That used to bother me, but about 10 years ago I passed the point where I needed to worry about whether I had a bachelor's or not."

Among the many projects Corey worked on in more than two decades at Kodak, he said one of the most notable was a parallel-processor array system for the Transputer, the groundbreaking processor built by the British firm INMOS. "It was just about the first microprocessor developed for parallel-processing use," said Corey. "We built these huge arrays of processors so they could crunch on images for motion picture digital editing in real time."

One problem was that at the time the Transputer was designed, in the 1980s, "we had no good way to get the data out to a display," he said. "So while the processing was going on in real time, the operator was waiting for display updates to come across the Sun VME bus. So I came up with a frame store architecture that was itself based on Transputers in conjunction with a large XC4000-series FPGA. It fit into the mesh of Transputers [designated] for communications, and then it delivered frame rate data out to a delivery device. We could configure huge arrays of these frame stores, with each one handling a portion of the display window, and then stitch the tiles all together in video on its way to the display." In this way, said Corey, "we could do real-time display of these really huge images that the motion picture people were working on."

Corey has also worked on the development of several digital-cinema projector proof-of-concept projects. In the early years, he did a lot of VME projects and created frame stores for both video and print applications. In the course of this varied work life, Corey has used every generation of Xilinx® FPGA, from the XC2064, Xilinx's first device, to the Virtex®-5.

"To many people, the FPGA has morphed into something that's really an ASIC that you can reconfigure," said Corey. "When they first came out, FPGAs weren't anything like an ASIC, but I've always found FPGAs to be devices that are great for rapid prototyping and low-volume manufacturing in all kinds of applications."

Although "a sizable part of the engineering segment is of the opinion that an FPGA is just a small ASIC," Corey perceives "some considerable differences, especially from the standpoint of how you design it. The penalty for making a mistake in an FPGA design is minor compared with making a mistake in a huge ASIC. For that reason, it is OK to take more risks when designing with an FPGA, especially if you are doing research projects or advanced development."

Corey was at Kodak roughly a year or so when he started using the XC2064. "That would have been in 1985 or 1986," he said. "After using that first FPGA, I got to the point where I never wanted to use hard logic again. I would go to great lengths to incorporate it straight into the FPGA. These days if I need DACs or A/D converters, I usually roll a sigma-delta implementation right into the FPGA. I've even done a phase-frequency comparator for a phase-frequency synthesizer in one of the old 3000- and then 4000-series parts."

As a longtime user of Xilinx FPGAs, he's witnessed firsthand the evolution of not only the devices but the methodologies to program them.

"Back in the days of the XC2064, you went into FPGA Editor and fiddled around by turning all the switches," said Corey. "There was no real front-end design software whatsoever, but thankfully that's not the case now. Today I prefer to drive the design process from a high level. You can cause a lot more bits to get toggled [that way]."

Over his many years in engineering, he has learned a number of programming languages, including Perl, Forth, C, Visual Basic and VHDL. "But I've always preferred GUI-based design, vs. hard coding," said Corey. "I think there still are a number of us in that arena." Nevertheless, Corey said that one of the many projects he's slated during his retirement is to become more familiar with Verilog. "I dabbled with Verilog a bit, but for the most part, I've been a VHDL user," said Corey, who notes that a fundamental part of being in the electronics engineering profession is a passion to learn new things, evolve your trade and embrace new technologies.

"It's one of the reasons FPGAs and I have become so fond of each other over the years," he said. "They keep evolving, too."

When he decided to retire in April, Corey penned a list of projects to keep himself busy. Not surprisingly for the consummate engineer, they all had some element of construction and building. He's currently doing a remodeling project on his home but said that while ticking off jobs done from the list, he's also been thinking of new electronics design projects and is strongly considering turning his "retirement" into merely a nice long vacation.

"I'm one of those people who is afflicted with the engineering disease," said Corey. "I really love engineering. It has to be the world's greatest job. Unfortunately, it's becoming more and more difficult to just be an engineer. A lot of companies want their engineers to become IP writers, and I find that stupefyingly boring. I'm a 'design it, simulate it, test it' sort of engineer. Let other folks write about the IP."

'The penalty for making a mistake in an FPGA design is minor compared with making a mistake in a huge ASIC. For that reason, it is OK to take more risks when designing with an FPGA.'

In particular, Corey said he's interested in further examining what he could do with systems that incorporate multiple soft cores. "The last few projects I was working on incorporated MicroBlaze™ processor cores inside a Virtex-5 using Platform Studio," said Corey. "I was doing some neat stuff with MicroBlaze, and so one of the things I want to do is experiment with using multiple PicoBlaze™ cores on an FPGA for some machine control ideas I have. Avnet makes a very nice, low-cost Spartan®-3A development board and I'm going to buy one of those."

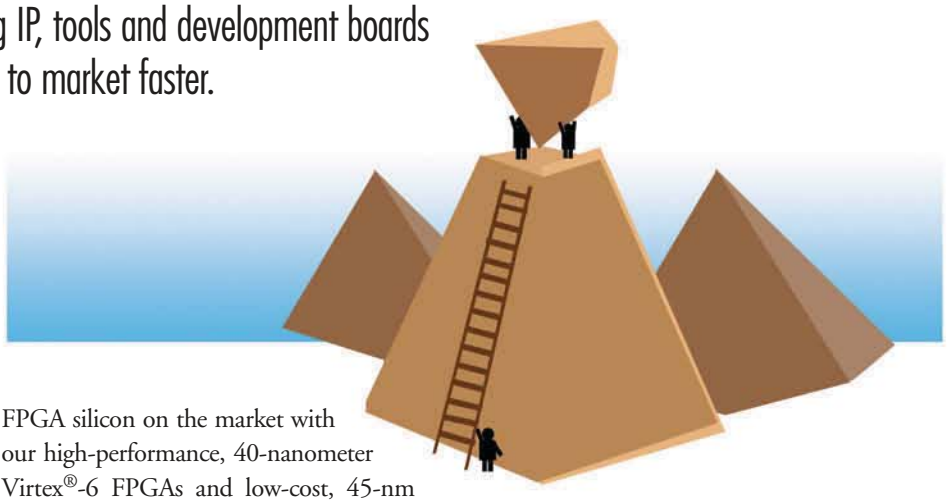
Asked if he has any other more-traditional retirement diversions in mind, Corey said he occasionally goes with friends to the shooting range but doesn't, at least yet, like the idea of spending a full afternoon doing the usual activities, such as golf or travel.

"Why would anyone waste an entire afternoon playing golf when they could be out building something?" said Corey.

# Targeted Design Platforms Take FPGA Innovation to New Heights

## New device families and accompanying IP, tools and development boards will help you get differentiated designs to market faster.

Frank Tornaghi
Senior Vice President
of Worldwide Sales
Xilinx, Inc.
*frank.tornaghi@xilinx.com*



Over my many years in the semiconductor business, including a long stint selling ASICs, I've witnessed firsthand the remarkable evolution of the FPGA from a neat little novelty device that ASIC teams used in the 11th hour of design projects for fixing problems in their ASICs, to a system-on-chip that today has a much stronger overall value proposition than any other logic device, ASIC and ASSP included.

Ever since Xilinx invented the first FPGA 25 years ago, FPGAs have radically grown in performance, capacity and functionality, displacing ASICs and ASSPs in a growing number of designs targeting the wired and wireless communications, automotive, ISM (industrial, scientific and medical) and aerospace-and-defense markets.

An ever-increasing number of designers are coming to realize that FPGAs offer the greatest mix of flexibility, time-to-market and overall cost savings of any logic device. If you make a mistake in your design or have to add last-minute functionality, you can simply modify your design, reprogram the FPGA and test it in your system immediately.

While the FPGA's value proposition over ASICs and ASSPs will only become more apparent to a wider number of customers as process technologies advance, Xilinx is making a concerted effort to further that advantage by launching Targeted Design Platforms. In addition to offering the best FPGA silicon on the market with our high-performance, 40-nanometer Virtex®-6 FPGAs and low-cost, 45-nm Spartan®-6 FPGAs, we're also offering you access to domain- and market-specific IP, tools, development boards and reference designs to help you get your differentiated designs to market quickly.

We created the Xilinx Targeted Design Platform concept to better serve your needs, so that if you are creating a design that's targeting a new standard or even entering a new market, the plan is to put all the elements in place to help you succeed.

If, for example, your next design is targeting the 40G/100G communications market, using our Targeted Design Platform you will not only have in your arsenal the most sophisticated, highest-transceiver-rate FPGAs on the market—offered in our 40-nm, Virtex-6 HXT devices, which have up to 64 transceivers, each operating at 11.2 Gbits/second—you will also have access to a comprehensive library of Xilinx and Xilinx-partner connectivity IP, providing you with key pieces of intellectual property required specifically for 40G/100G communications designs.

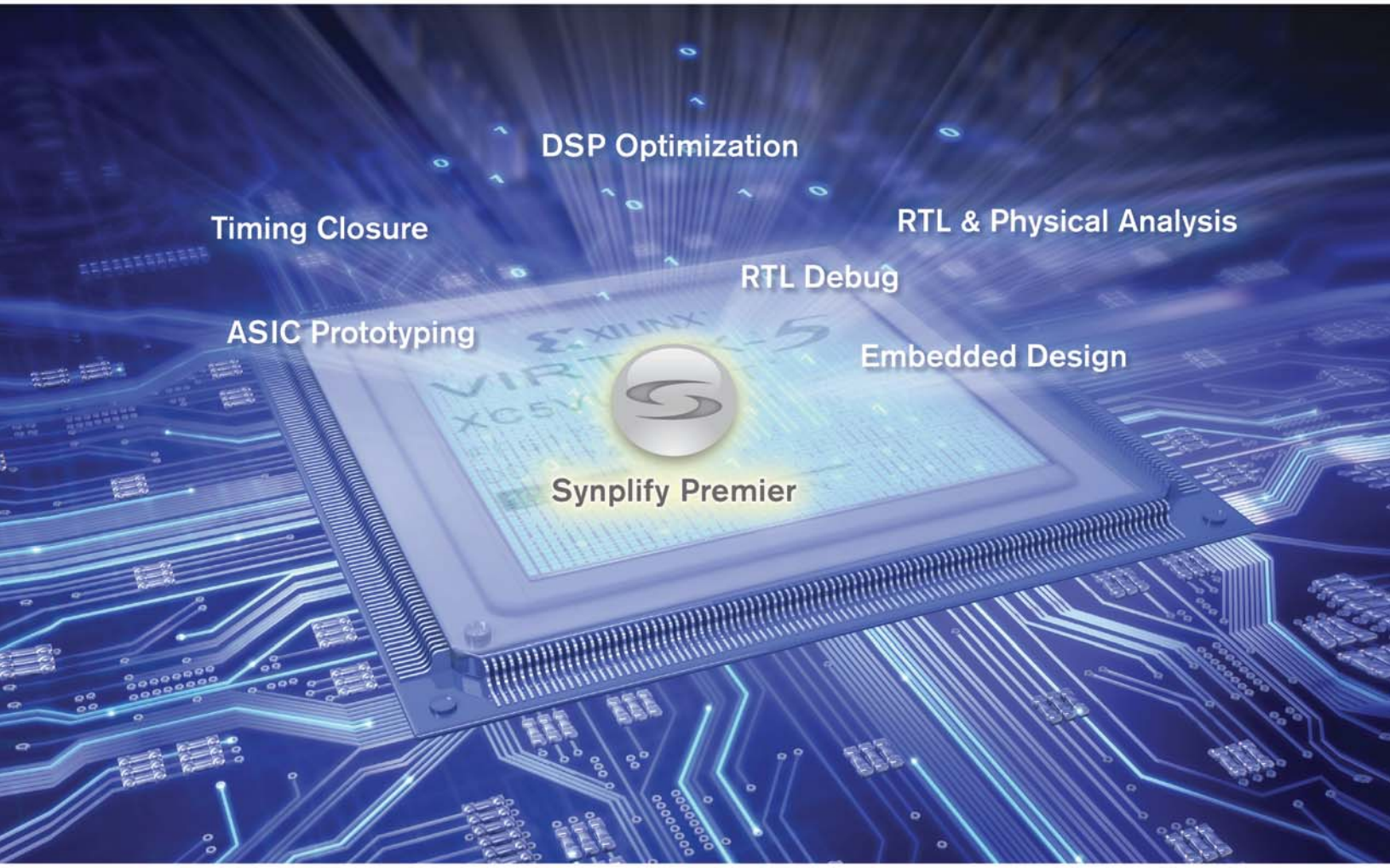
You will be able to implement this market-specific IP, our domain-specific platform IP and our base platform IP to quickly assemble a high percentage of your design, so that you can focus your efforts on the veneer of differentiation to maximize your success in your market. What's more, we offer a family of evaluation kits (base, domain- and market-specific kits) and reference designs to help you get up and designing quickly.

If you are an engineer in the embedded-software domain, we offer the domain-specific tools as well as microprocessor and subsystem IP you'll need to quickly produce embedded innovations in an environment that's familiar to you. Xilinx remains committed to embedded processing, leveraging our current family of microprocessor cores, with more road map announcements to come very soon.

If you are an algorithm developer in the DSP domain, we offer the domain-specific tools and IP you'll need to implement your algorithms in our DSP-slice-rich Spartan-6 and Virtex-6 FPGAs. What's more, you will do so in an environment that's familiar to you.

In fact, I believe every single one of our customers will see great benefits and reach new levels of productivity with our Targeted Design Platform offerings. Over its 25 years of existence, Xilinx has taken great pride in working with you, our customers, in creating new, exciting innovations. With Targeted Design Platforms, I believe we can take your innovations to new heights.

# Accelerate FPGA Design




## Synplify Premier, the Ultimate in FPGA Implementation

The Synplify® Premier software from Synopsys® is the preferred synthesis and debug environment for complex FPGA designs. It provides a comprehensive suite of tools and technologies for advanced FPGA designers as well as ASIC prototypers targeting a single FPGA. The Synplify Premier solution addresses the biggest FPGA design challenges including timing-closure, logic verification, IP support, ASIC compatibility, DSP implementation, and RTL debug while providing tight integration with Xilinx® back-end tools. Synplify Premier supports DesignWare®components and performs detailed logic placement which is passed on to the Xilinx router for final implementation. With final placement knowledge during synthesis, design iterations are significantly reduced resulting in shorter development schedules.

To learn more about how the Synplify Premier software can help you achieve your design goals, visit
http://www.synplicity.com/synplifypremier/xcell_premier.html



Synplicity®
Simply Better Results

SYNOPSYS®
Predictable Success

XILINX

# RISE ABOVE

**New Xilinx Targeted Design Platforms give designers a boost to take innovation to a much higher level.** These comprehensive platforms combine Virtex®-6 or Spartan®-6 devices, the ISE Design Suite 11, development hardware, IP, reference designs, documentation, and service and support—so you're way ahead of the competition before you even start. Plus, everything works together seamlessly, so design teams can focus on product differentiation, add more value, and make the whole project more successful. Learn more now at **www.xilinx.com/6**.



PN 2400