A Reconfigurable Platform for Multi-Service Edge Routers

Christoforos Kachris, Stamatis Vassiliadis
Computer Engineering Department
Delft University of Technology
The Netherlands
{kachris, stamatis}@ce.et.tudelft.nl

ABSTRACT

A main feature of current FPGAs is that they can be dynamically reconfigured to meet the network traffic requirements. In this paper we present a case study for a multi-service edge router in which the number of processors and co-processors is dynamically reconfigured to meet the network traffic workload. The system targets the Xilinx Virtex4 FPGA platform and uses the MicroBlaze soft processors for header processing and hardware acceleration units for payload processing. Furthermore, two schemes are compared for the reconfiguration of the system. The first one has fast response time but is prone to network burst traffic while the second one has slower response time but is more robust to burst traffic. The performance evaluation shows that the reconfigurable platform can achieve up to 1.5x speedup compared to a static system.

Categories and Subject Descriptors

B.7.1. **[Types and Design Styles]**: Gate Arrays, C.1.3. **[Other Architecture Styles]**: Adaptable Architectures, C.2.6. **[Internetworking]**: Routers

General Terms

Performance, Design

Keywords

Reconfigurable logic, FPGA, Edge routers

1. INTRODUCTION

The packet's processing requirements in the current network have been increased significantly the last years. To face the increased bandwidth and the processing requirements, more sophisticated network devices have to be used that are based on application specific processors and not on general-purpose processors. The most widely used platforms are the network processors, the application specific integrated circuits (ASIC) and the Field-Programmable Gate Arrays (FPGAs) [1]. The network processors usually accommodate multiple RISC processors that have been optimized for network processing. The use of processors for the packet manipulation provides high flexibility but usually has low performance. The ASICs, on the other hand, use hardwired units to process the packets, therefore provide high performance but they lack of flexibility. Furthermore, the development of ASICs using the current technologies has become very expensive due to the increased Non-Recurring Engineering (NRE) cost. The FPGAs provide a promising alternative mainly because they can provide a fair combination of the flexibility of the network processors and the performance of an ASIC. The flexibility is preserved through the use of embedded RISC processors into the FPGA and the fact that the FPGAs can be reconfigured (similar to software upgrade) while the performance is preserved using hardware acceleration units to perform the payload processing.

Furthermore, a major issue in the design of network device is the power limitations. Network processors use multiple instances of processors running at high clock frequencies to face the processing requirements. Hence, the power consumption of theses devices is usually high. The use of FPGAs can lower the power consumption since a lot of functions can be implemented in hardware that is more efficient than software in terms of performance and power consumption. Moreover, as the size of the transistors shrink, the static power starts to hold a significant portion of the total power consumption. Hence, in the future a major challenge will be how to reduce the static power consumption. The nature of network traffic is quite dynamic and unpredictable. In the case of multi-service router each packet belongs to a specific flow and each flow has different processing requirements. Some of the flows needs only header processing while others need also payload processing, such as encryption or compression. Therefore the utilization of the processors can be high and the utilization of the hardware accelerators units can be low in some cases and vice versa. Therefore, the current network processors are large devices that host many processors and hardware units in order to face worst case network traffic resulting in high static power consumption. The use of dynamically reconfigurable FPGAs can face this problem. Instead of containing all of the modules for every workload traffic distribution, a part of the FPGA can be adapted to the optimal configuration for the specific network traffic which results to smaller devices, hence to lower static power consumption.

In this paper we present a representative case study in which the FPGA can be used to adapt to the network traffic. The FPGA platform can be used in a multi-service edge router. The multi-service edge routers need to process several packets that may belong to different flows. Many flows need just forwarding of the packets. Some other flows need encryption or decryption of the packets if they belong to secured traffic (such as VPN networks). And some other packets may need compression or decompression if they belong to flows of wireless networks. This paper presents how the use of dynamically reconfigurable FPGAs can be efficiently deployed to implement such as a system that can provide high performance and low power consumption. The main contribution of this paper is the following:

- The design of a multi-service edge router in a reconfigurable logic platform that change the number of processors and the number and type of co-processors to meet the network traffic
- The performance evaluation of the reconfigurable platform using two schemes for the reconfigurable manager depending on the traffics (burst traffic or steady traffic)

The paper is organized in the following way. Section 2 presents the related work in the area of adaptive system for network processing both for network processors and FPGAs. Section 3 presents the FPGA architecture of the proposed scheme and two schemes for the implementation of the reconfigurable manager. Section 4 presents the performance evaluation of the system for several network traffics and the comparison of the proposed schemes for the reconfigurable manager. Finally, Section 5 presents the conclusions and the future work.

2. RELATED WORK

In the area of network processors, a case for run-time adaptation in packet processing systems is presented in [2], implemented using the Intel network processors. The proposed framework is used to allocate several micro-engines (simplified processors) depending on the fluctuation of the network flow. Each micro-engine is used for separate flow, thus the system is used to automatically allocate a specific number of processor for each flow. The number of processors allocated to each flow depends on some thresholds on the FIFO occupancy of each flow.

In the area of reconfigurable platform, several research papers try to address the exploitation of dynamic reconfiguration network processing applications. In [3] [4], a reconfigurable programmable router has been introduced that is mainly used in active networks. Active networks are networks in which the packets are processed by emerging protocols that are either included into the packet or can be downloaded dynamically into the router. The system consists of general purpose CPUs and hardware plug-ins. Each plugin has an SRAM and an SDRAM interface to communicate with the memory, and a custom interface to a 32-bit wide ring in order to communicate with the CPUs and the other plugins. The plugins are dynamically configured kernel modules used to process the active packets, that can be downloaded by a trusted server. The system has been implemented into two FPGAs, one used as the network interface device and one used as the host of the hardware plug-ins.

In [5], a reconfigurable system called Programmable Protocol Processing Pipeline (P4) has been introduced. In this case, a set of FPGAs is used in a pipeline way in order to accelerate the packets processing. Every device has a FIFO buffer associated with it that is used to load and store the processed packets. The devices are connected using a switching array that can include or exclude processing elements. As an example a Forward Error Correction (FEC) is used as a protocol processing function. Although, the FPGAs are able to be reconfigured dynamically, in that paper there is only the performance evaluation of a static design, and not of a dynamically reconfigurable device.

In [6], a reconfigurable network coprocessor platform is presented called DynaCore. In this case a platform mapped into an FPGA is presented that can accommodate hardware accelerator units. The platform includes a dispatcher that is used to send the incoming packets to the hardware acceleration units. The system consists only of hardware acceleration units, and a connection of the hardware units with the general purpose processing elements used for the remaining header processing is not presented.

The design of the reconfigurable controller that must hide the reconfiguration overhead has also been addresses in many research papers. In [7], a reconfiguration manager is presented to hide the reconfiguration latency. The manager applies two different techniques at run-time: prefetch scheduling and replacement. In the prefetch scheduling technique, the manager schedules the reconfiguration based on scheduled sequence of tasks and their loading latency. Furthermore, they apply an intertask optimization technique to further decrease the reconfiguration overhead. In [8] various types of prefetching to reduce the reconfiguration overhead are also applied for the configuration manager. They present techniques such as static, dynamic and hybrid configuration prefetching. Furthermore, the configuration manager applies relocation and defragmentation techniques to reduce the dynamic reconfiguration overhead. A similar approach is presented in the MOLEN framework [9], in which a polymorphic processor is presented incorporating both general purpose and custom computing processing. The reconfiguration is mainly scheduled by the software, in which the hardware accelerators are pre-loaded in order to decrease the reconfiguration overhead.

In [10] a framework for reconfigurable computing scheduling is presented in which the main task is the scheduling of the reconfigurable units at design time. A similar approach is also presented in [11][12], in which the optimum scheduling sequence is investigated based on the task graphs of the applications. In all of these approaches the scheduler can decide based on the task graph the sequence of the reconfiguration. On the other hand, in the area of network processing the workload (network traffic) is quite dynamic therefore the management of the reconfigurable units should be performed at run-time. Furthermore, the scheduling of the reconfigurable units should be simple enough so there is no much processing overhead. Hence, complicated algorithm should be avoided that will consume part of the processing power.

3. RECONFIGURABLE ARCHITECTURE

A typical multi-service edge router has to process packets that belong to several flows. Each flow can have its own processing requirements. In our case, we present a case study in which the router has to process three different flows. The packets that belong to the first flow are forwarded using a queue scheduling scheme (Deficit Round Robin - DRR), therefore only some header processing (IP forward) and some queue management is required that is performed by the processors. The packets that belong to the second flow are used in secure connections (such as Virtual Private Networks - VPN) thus the packets besides header processing need also payload processing that is performed by specialized hardware co-processors (DES co-processors). Finally, the packets that belong to the third flow are used in wireless connections that need compression to reduce the amount of data.

In this case, a hardware acceleration unit is used to compress the payload of the data (Lempel-Ziv Compression- LZC).

3.1 System Architecture

The platform has been implemented in the Xilinx Virtex4 FPGA platform. The processors and the co-processors are attached to a 32-bit wide shared bus (OPB Bus). The architecture of the reconfigurable platform is shown in Figure 1. The incoming packets are stored into FIFO_IN and are forwarded to the pool of the processors. The packets are classified and depending on the flows that they belong to, they are forwarded to the corresponding FIFO. If the packets belong to secured connections (such as VPN network) the packet are forwarded to FIFO_DES in which the DES co-processors are used to encrypt or decrypt the data. If the packets belong to a connection with a wireless device, the packets are forwarded to FIFO_LZC in which the compress units are used to compress or decompress the packet payload. Finally, if the packets do not belong to these flows they are just forwarded to FIFO_OUT after the DRR scheduling and the IP Forwarding (header modifications, checksum re-calculation, etc). A Reconfigurable Management Unit (RMU) is used to reconfigure the platform depending on the network traffic.

The bottleneck of the system varies as the network traffic changes in the edge router. Figure 2 shows the number of cycles that are required for each unit (processors, encryption co-processor, and compressor co-processor) for several flow distributions. As it is shown, when the majority of the packets need just forwarding (60%) then the required number of cycles used by the processors is much more than the required cycle for the co-processors. In case that the majority of the packets belong to a secure connection then the required number of cycles from the DES unit is much more than the cycles for the processors and the compression unit. The same behavior is also present in the case that the majority of the packets need compression. Most of the cycles are used by the compression co-processors.

Therefore, in order to create a balanced system in which all of the processors and the co-processors are equally utilized depending on the network flow distribution, an adaptive system is required. Based on the processing requirements and the processing capabilities of the modules, three configurations are used as it is shown in Table 1. The system consists of a static part and a reconfigurable part. In the static past two processors, and two co-processors for each payload processing are used. The reconfigurable part accommodates two of the units (either two processors, or two DES units or two LZ compression units) depending on the configuration. The reconfiguration in which the two co-processors must be replaced by other co-processors is straightforward. The reconfiguration management unit (RMU) waits until the processing of the current packets has finished and then the co-processors are replaced by the new modules. In the case that the units must be replaced by processors the task is more complex. The RMU must replace the modules by processors, and then transfer the program to the local memory of the processor before it is initialized. The partial reconfiguration file contains the initialization memory of the internal processor's memory units (Block RAMs), hence the total time is just the time to reconfigure the device plus the time to initialize the processor. The maximum number of slices of the reconfigurable area is 4438 slices (96 rows x 7 columns of the FPGA, ¼ of the total area). This area can accommodate either 2 processors or 2 co-processors. Using the Xilinx ICAP module for the reconfiguration running at 100MHz with 32 bit wide bus the reconfiguration time is almost 2.5ms. The time to initialize the processor is negligible, since no operating system is used in the processors (network processing functions mainly do data manipulation, hence no operating system is required).

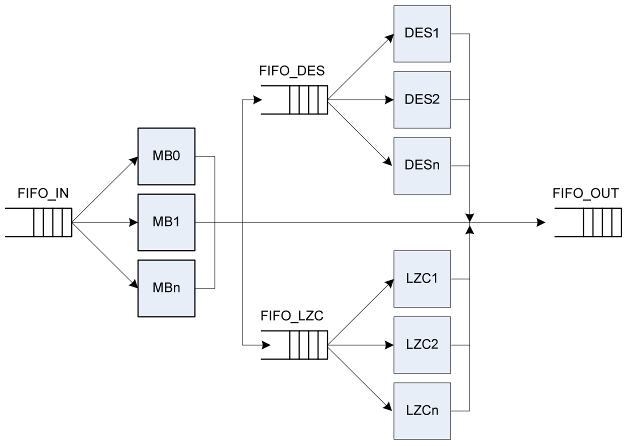

Figure 1. Reconfigurable Architecture

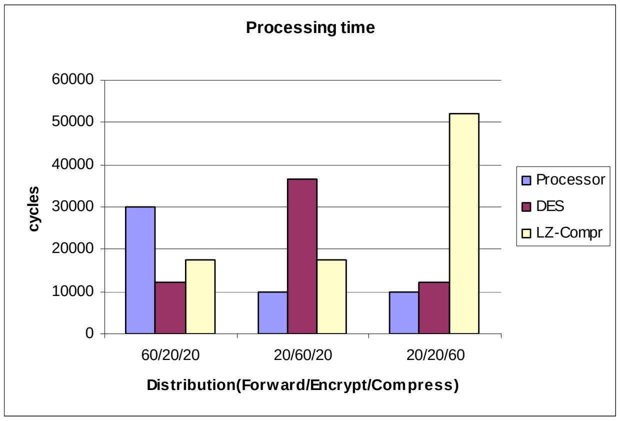

Figure 2. Bottleneck of the platform for several distributions

Table 1. Configurations

Configuration	# of Processors	# of DES	# of LZC
Config_A	4	2	2
Config_B	2	4	2
Config_C	2	2	4
Reconfiguration	2	2	2

3.2 Reconfigurable Manager

The reconfigurable manager unit (RMU) can be implemented using two different schemes. The first option is to measure the distribution of the processed packets. After the classification, each processor increases a specific counter depending on the flow that the packet belongs to. These counters can be accessed by the RMU to decide if it will perform a reconfiguration or not. The decision is based on these counters. If the percentage of packets that belong to the specific flow reaches a specific threshold (e.g. 50%) then the RMU reconfigures the platform to accommodate two more processors or co-processors for the specific flow. This scheme can provide fast response time to network traffic changes but is prone to burst traffic that do not last long.

The second scheme performs the reconfiguration based on the occupancy of the FIFOs. In this case the platform remains in the same configuration until the occupancy of one of the FIFOs (FIFO_DES or FIFO_LZC) reaches a certain threshold. Based on these thresholds the manager selects one of the configurations from Table 1. In this case, the platform is reconfigured to remove two units (either two processors or two co-processors of the other flow) and two co-processors or processors for the specific flow are added. This scheme has slower response time to network traffic changes but is more robust to burst traffics that do not last long as it is shown in the performance evaluation.

4. PERFORMANCE EVALUATION

In order to evaluate the system an initial implementation of the system using the Xilinx EDK 8.2 platform has been used. Using the post place and route simulation model we measured the performance of the system for several configurations.

Table 2 shows the number of cycles that are required by the MicroBlaze processor for the Deficit Round Robin (DRR) scheduling (from the the CommBench benchmark [13]) and the IP forward. Table 3 shows the performance of the hardware accelerators. The accelerators are modified version from the Opencores [14]. Table 4 shows the execution time to process 100 packets (the packet size is set to 512 bytes) for several network flow distributions (Forward/Encrypt/Compress) by the 3 configurations (the bold number shows the minimum execution time for each traffic distribution). The last configuration is the one in which the system is performing the reconfiguration. In this configuration the reconfigurable part cannot be used therefore the system consists of 2 processors, 2 co-processors for encryption and 2 co-processors for the compression. As it is shown in the table in this case the processing time is the maximum of the other three columns. Therefore, the reconfigurable management unit (RMU) must perform the reconfiguration in a way that the reconfiguration overhead can be overcome by the benefit of using the best configuration in each network traffic distribution.

Table 2. Processor performance for several applications

Processor Performance	Cycles
IP Forward	303
DRR[2]	1105

[2]DRR for 20 queues

Table 3. Performance of the Co-processors

Module	Performance
DES encrypt	400Mbits/sec
LZ compress	320Mbits/sec

Table 4. Processing time (usec) for several configurations

Distribution	Config_A	Config_B	Config_C	Reconfiguration
60/20/20	**184**	356	356	356
20/60/20	311	**197**	321	321
20/20/60	390	400	**212**	400

In order to speedup the performance evaluation, we created simulation models for the processor and the co-processors using the post place and route simulation results. The program of the processors is explicit parallel, since each processor performs the same function for different packets. Hence, the communication overhead of the parallel processors is negligible. To measure the speedup of the system compared to a static system, we created 2 types of network traffic. In the first case, the network flow distribution was remaining the same for a certain amount of time (network stability). In the second case, we inserted small amount of packets in which the distribution was different from the current distribution. That way we could measure the performance of the system for bursty traffic. Figure 3 shows the speedup of the system compared to a static system (in which 4 processors and 2 co-processors for each flow are used (Config_A). The speedup is calculated using the following equation.

$$speedup = \frac{ExecutionTime_{Static}}{ExecutionTime_{Dynamic}}$$

Figure 3a shows the speedup when the network traffic does not have bursty traffic. Thus, the network flow distribution remains the same for a certain time (network stability) within some small fluctuations +-5%. As it is shown in the figure, the RMU that reconfigures the system using the flow distribution of the packets is always better than the RMU that used the FIFO occupancy to perform the reconfiguration. This is due to the fact that the distribution scheme has better response time, since it can recognize the network distribution at processing time. When the network traffic is unstable, (e.g., the network stability is 5 ms) then the FIFO-based RMU has a worse execution time than a static system, while the distribution-based RMU is slightly better than static system. The worse execution time of the FIFO-based RMU is due to the fact that the system remains in the reconfiguration state the majority of the time, thus the overall performance is worse than the static system. When the network traffic becomes more stable (network stability is 20ms) then both of the RMUs perform a significant speedup compared with the static system but still the Distribution-based RMU outperforms the FIFO-based RMU. Finally, when the network becomes more stable (200ms) then in this case the two schemes have almost the same speedup.

On the other hand, when the network traffic becomes unstable (with bursty traffic) then the FIFO-based RMU is better than the distribution-based RMU. Figure 3b shows the performance of the system when the network traffic is stable for a specific time (network stability) but it is interrupted by fragments of time in which the distribution change for a while and then returns to the previous distribution. Figure 3b shows the speedup when 2 spikes are inserted into the network traffic with 2ms duration.

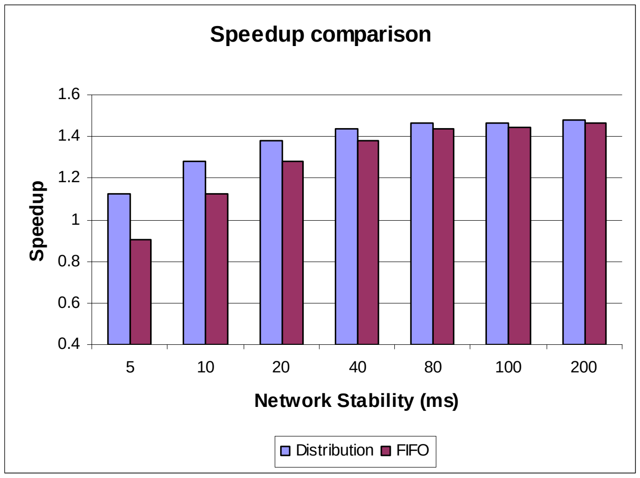

Figure 3a. Speedup comparison

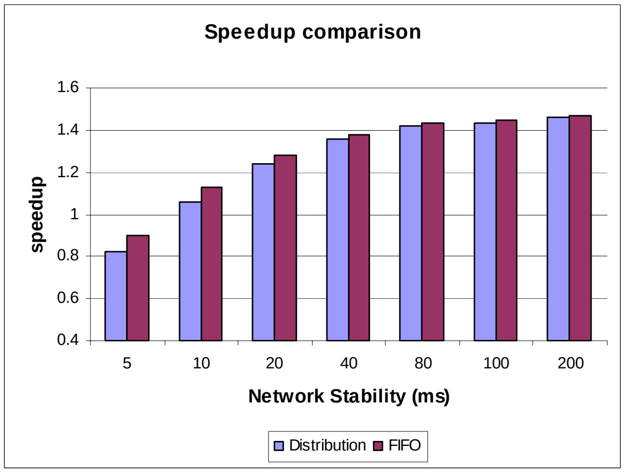

Figure 3b. Speedup comparison in bursty traffic

Figure 4 depicts the comparison of the distribution-based RMU and the FIFO-based RMU for several numbers of burst spikes. As it is shown in the case of one spike in every network stability period the speedup is the same for both schemes. As the number of spikes increase the FIFO RMU performs better, since it can absorb the spikes. Therefore, a dynamic system could be used in which a module could capture the network stability and activate the appropriate RMU unit. In case that the network stability is small (less than 10 ms) then the RMU can keep the system to same configuration to avoid the overhead. In case, that the network stability is high enough then the system can use either the Distribution RMU or the FIFO RMU depending on the number of network spikes.

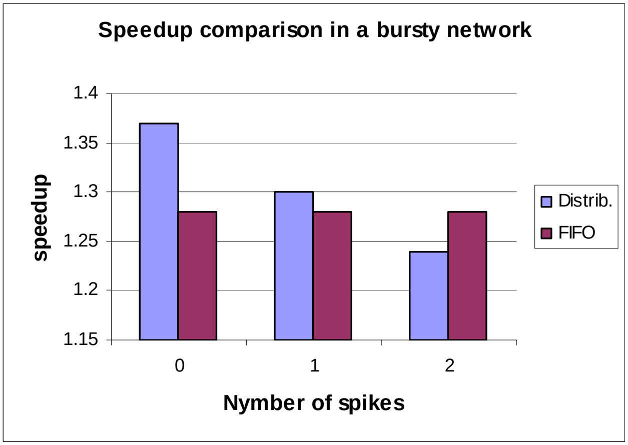

Figure 4. Speedup comparison in bursty network for 20 ms

5. CONCLUSIONS

As it is shown in this paper the use of dynamic reconfiguration in network devices can significantly improve the performance of the system by adapting it to the workload. Furthermore, through the dynamic reconfiguration the system can be implemented in smaller devices, thus reducing the cost and the static power consumption of the chip. In addition, two reconfigurable schemes have been presented that can be used to control the reconfigurable parts depending on the network features (traffic fluctuation). The RMU that use the current packet flow distribution has higher response time and is better for networks that are more stable, while the RMU that use the FIFO occupancy is better for networks that have bursty traffic.

6. ACKNOWLEDGMENTS

This work was supported by Sandbridge Technologies Inc.

7. REFERENCES

[1] R. Warden, Design Considerations for Edge Router, Online article, www.commsdesign.com, 8 May 2002

[2] A. Raghunath, A. Kunze, E. J. Johnson, V. Balakrishnan, Framework for supporting multi-service edge packet processing on network processors, in *Proceedings of the ACM First Symposium on Architectures for Networking and Communications Systems*, Princeton, NJ, October, 2005

[3] J. Lockwood, N. Naufel, J. Turner, D. Taylor, Reprogrammable Network Packet Processing on the Field Programmable Port Extender (FPX), in *Proceeding of the International Symposium on Field Programmable Gate Arrays* (FPGA'01), Monterey, CA, February 2001

[4] D. Taylor, J. Turner, J. Lockwood, "Dynamic Hardware Plugins (DHP): Exploiting Reconfigurable Hardware for High-Performance Programmable Routers," *Computer Networks*, vol. 38, no. 3, pp. 295-310, February 2002

[5] I. Hadzic, W. Marcus, and J. Smith, On-the-fly Programmable Hardware for Networks, in *Proceedings of the IEEE Global Communications Conference (GLOBECOM98)*, Sydney Australia, November 1998

[6] J. Foag, R. Koch, Architecture Conception of a Reconfigurable Network Coprocessor Platform (DynaCore) for Flexible Task Offloading, in *Proceedings of the Advanced Networking and Communications Hardware Workshop (ANCHOR 2004)*, Munich, June 2004

[7] J. Resano, D. Mozos, F. Catthoor, D. Verkest, A Reconfiguration Manager for Dynamically Reconfigurable Hardware, *IEEE Design and Test of Computers*, Sept.-Oct. 2005, vol. 22, no. 5, pp. 452- 460

[8] Z. Li, S. Hauck, Configuration Prefetching Techniques for Partial Reconfigurable Coprocessor with Relocation and Defragmentation, in *Proceedings of the Tenth ACM International Symposium on FPGAs*, Monterey, CA , April 2002

[9] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K.L.M. Bertels, G.K. Kuzmanov, E. Moscu Panainte, The Molen Polymorphic Processor, *IEEE Transactions on Computers*, pp. 1363- 1375, November 2004, vol. 53, no. 11

[10] R. Maestre et al., A Framework for Reconfigurable Computing: Task Scheduling and Context Management, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 6, December 2001

[11] K. Papademetriou, A. Dollas, Performance Evaluation of a Preloading Model in Dynamically Reconfigurable Processors, in *Proceedings of the IEEE International Conference on Field Programmable Logic and Applications (FPL'06)*, Madrid, Spain, August 2006

[12] K. Papademetriou, A. Dollas, A Task Graph Approach for Efficient Exploitation of Reconfiguration in Dynamically Reconfigurable Systems, in *Proceedings of the IEEE Symposium on Field Programmable Custom Computing Machines (FCCM'06)*, San Jose, CA, April 2006

[13] T. Wolf, M. Franklin, CommBench-a telecommunications benchmark for network processors, in *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software*, Austin, TX, USA, 2000

[14] Basic DES Crypto Core, OpenCores, www.opencores.com