Analysis of a Reconfigurable Network Processor

Christoforos Kachris, Stamatis Vassiliadis

Computer Engineering Lab
Department of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
The Netherlands
{kachris, stamatis}@ce.et.tudelft.nl

Abstract

In this paper an analysis of a dynamically reconfigurable processor is presented. The network processor incorporates a processor and a number of co-processors that can be connected to the processor either directly or using a shared bus. The analysis investigates the configuration (in terms of co-processor distributions and interface), formulates the throughput that meets the network demands and the constraints of the platform (area, bus bandwidth, etc.) and takes into account the reconfiguration overhead. To find the configuration that meets the constraints, the platform is formulated into integer linear programming equations. Furthermore, the results of two case studies are presented, for a soft- and a hard- IP core processor, that uses three flows with different processing requirements (IP forward, encryption and media processing). In each case the number and the type of co-processors is shown in terms of the network distribution and the average packet size. Finally, the mapping of the framework in the Xilinx FPGA platform is discussed.

1. Introduction

The increase of the Internet bandwidth has created the need for more powerful processors, specifically designed for packet processing; the network processors. Network processors combine the flexibility of the general-purpose processors with the increased performance of the application specific integrated circuits (ASICs). Network processors have been evolved from simple TCP/IP accelerators to complete platforms able to process million of packets in core and access networks. The first network processors were used only for the processing of the lower OSI layers such as layer 2 or layer 3, while now some network processors are used even for layer 7 processing such as XML processors [7] and the web balancing processors. The architectures of the network processors (NP) vary from dataflow architectures such as the network processor from Xelerated [1] to multi-processors multi-threaded platforms such as the IXP2400 from Intel [2]. Many functions of a network processor, that are extremely demanding, have been implemented in district chips and are used as the co-processors. Some examples of these processors are Queue Managers, Traffic, Managers, Network Encryption processors, Intrusion detection processors etc. FPGAs with embedded processors are ideal platform candidates for the development of network processors that can be both flexible and high performance.

The main challenge in network processors is to find the right micro-architecture for the targeting network workload. The network processors that are targeting core routers must be able to process million of packets per second but the processing requirements for each packet are not so demanding; usually the processing is a just an IP forwarding. On the other hand, the network processors that are targeting access networks or residential gateways have to process less packets but each packet has demanding processing requirements such as fragmentation and reassembly, encryption, transcoding etc.

Furthermore, the traffic in the Internet varies from simple small packets used by Instant Message Services (IMS) to large packets used for media applications such as video and audio streams. Each application with different processing demands can be categorized into different flows. The distribution of the flows in time can vary significantly. For example, in a main server the majority of the packets in the working hours can be packets that need encryption processing based on the fact that many on-line transaction are processed in working hours. On the other-hand, media processing or transfer of media packets that need transcoding are usually take place in non-working hours. In [13] and [15] there is a detailed analysis of how the network traffic changes during the day or the week. The changes are in terms of the application, the

1-4244-0054-6/06/$20.00

protocol and the size of the packets. Hence, it is very difficult to design a network processor that will be able to efficiently process the changing networks flows. Usually, there is a range of processors, each one for different topology (core networks, access networks) that can meet the constraints of each topology, but none of them is optimal designed for each network workload.

The main goal of this paper is to formulate a dynamically reconfigurable network processor that can be adapted to the network demands by using the co-processor in the most efficient way. Hence the main contribution of this paper is:

- a formulation for the throughput of a dynamically reconfigurable network processor
- a method to find a configuration that meets the network demands and a limited number of platform constraints (area, bus bandwidth, etc.) and takes into account the reconfiguration overhead
- two case studies for a soft- and a hard-IP core processor with different workload demands on several average packet sizes
- a mapping of the dynamically reconfigurable network processor into a Xilinx FPGA platform

In Section 2 we present the related work in design space exploration of network processors. Section 3 presents the architecture framework of the system. Section 4 presents the analysis of the system and the formulation, while section 5 presents the experimental results for a case study with three different workloads for two processors and the mapping into an FPGA. Finally, section 6 presents the conclusions and the future work.

2. Related work

The use of design space exploration (DSE) tools can be very useful when designing the micro-architecture of the network processor. EXPO [3] is a DSE tool that uses the theory of the arrival and service curves to model the operation of a network processor. The computation complexity in this case is too expensive, thus they use a piecewise linear approximation of all arrival and service curves. The network processors can be modeled in a task graph and given the mapping of tasks to available resources it can estimate the Pareto-optimal solution for access and backbone networks. The tool is restricted to model a system with a common bus that every resource is attached to this bus. In [4, 12] an automated exploration framework is developed that is used in a soft multi-processors platform. The application is modeled as a task graph and each task is allocated in one of the processors. The tool is used to find the optimum partitioning of the IPv4 packet forward application into an array of processors, but this tool does not incorporate co-processors which are essential parts in network processors. In [5] a design space exploration is performed using several parameters of a general-purpose processor such as the processor clock rate, the instruction and data cache size, the area and the memory access time. The CommBench benchmark [23] is used to illustrate the difference of the optimum configuration using packets that only need header processing versus packet that need also payload processing. The model is applied both to a single processor and multiple processors. In [24] a design space exploration is performed for network programs on different architectures. The compared architectures are a speculative super-scalar processor, a fine-grained multithreaded processor, a single chip multiprocessor and a simultaneous multithreaded processor (SMT). The benchmark that it was used includes IP forward and MD5 and DES encryption processing. In [18] a design space exploration of the System-On-a-Chip (SoC) communication of the components is performed. The number of busses and bridges are investigated in order to find the optimum configuration for a given graph of connected modules. In [8, 9] a platform has been developed that use hardware plug-ins to accelerate the performance of a programmable router. The hardware modules are connected with external SRAM and DRAMs memories. The hardware plug-ins are allocated in a separate chip from the one that incorporates the processing elements. Furthermore, there is not a design space exploration thus it is quite difficult to find the configuration that meets the network and application demands. Finally, STMicroelectronics has presented a system-level exploration platform for Network processors called StepNP in [25, 26]. In that case the platform contains multi-threaded processors connected with a custom network-on-a-chip. The system is modeled at the functional and transaction levels and not at a cycle-accurate level.

In our case the design space exploration targets a single chip dynamically reconfigurable architecture that can be adapted to meet the workload of the network. The framework incorporates a processor with a library of co-processors that can be connected to the processor either directly or using a shared bus in order to find a high-throughput configuration that meets the constraints and in addition taking into account the configuration overhead.

3. Reconfigurable Network Processor Framework

The reconfigurable network processor architecture is targeting the Xilinx Platform. In this work, two frameworks have been designed and analyzed. In the first case we used a soft-core processor and in the second case a hard-core processor. An analysis of the performance and the constraints is presented for both cases. In the first case

the MicroBlaze soft-core processor has been used augmented with co-processors connected with a direct connection and a bus with the processor. The direct connection is called Fast Simplex Link (FSL) [19] and can be used for the direct communication of the register's value with the co-processors. The maximum number of FSL interfaces that are supported by the MicroBlaze is eight. The shared bus that it was used was the On-chip-Peripheral-Bus (OPB) [21] which is able to sustain 500Mbytes/sec throughput. The MicroBlaze is also connected with one block of RAM for Instruction and Data Memory using the Local Memory Bus (LMB), while it is also connected, through the bus, to one block of larger RAMs that is used as the source RAM of the packets. The network interfaces are not shown since the main motivation for this system is the exploration of the network processing. An additional block RAM is also attached to the OPB bus and it is used as a small IPv4 Forwarding Table. The block diagram of the architecture that it was used as an experimental platform is shown in Fig. 1.

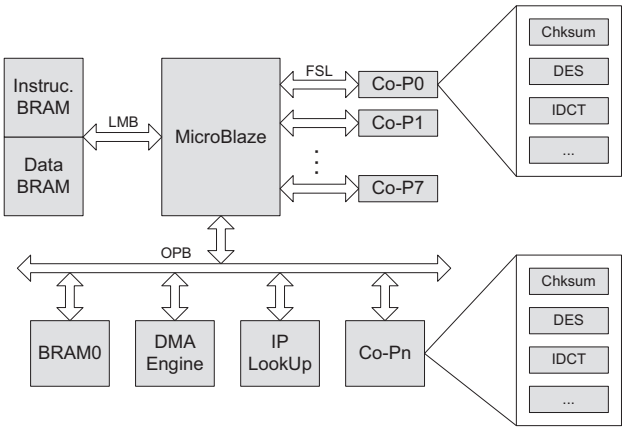

Figure 1. The MicroBlaze framework

In the second case, the hard-core processor PowerPC is used augmented again with co-processors communicating in a direct interface and in a bus with the processor. The direct interface is the Auxiliary-Processor-Unit (APU) [20] connected with the Fabric Coprocessor Module (FCM) which is similar to [14] and can be used to transfer the register's value directly to the co-processors. The shared bus that has been used in this case is the Processor-Local-Bus (PLB) [21] which is able to sustain 1600Mbytes/sec. The PowerPC is also connected to a block of RAM for Instructions and Data Memory using the On-Chip-Memory (OCP) interface. An additional block RAM is also attached to the PLB bus and it is used as a small IPv4 Forwarding Table. The block diagram of the architecture that it was used as the second experimental framework is shown in Fig. 2.

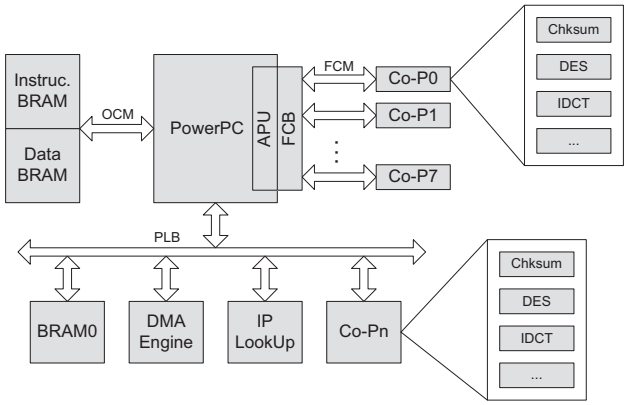

Figure 2. The PowerPC Framework

In the current design three modules have been used as co-processors. A checksum hardware block is used for the calculation of the header's checksum, a Data Encryption Standard (DES) unit for the processing of encrypted data and an Inverted Discrete Cosine Transformation (IDCT) unit for the transcoding of the payload. Although the transcoding needs more units for the processing such as variable length encoders, the use of IDCT is a major module to accelerate the transcoding.

4. Analysis of the Reconfigurable Network Processor

The main goal of a network processor designer is to find the optimum micro-architecture for a specific workload given specific constraints. The constraints can be in aspects of power, energy, performance or area (or usually a combination of these). In this paper a formulation of a reconfigurable network processor is presented in order to find a configuration that meets the network workloads distribution and other constraints. The ability of the network processors to adapt itself to the different workload could increase the performance of the processors by using co-processors that are useful for the majority of the time. In this work, three different flows have been used with different processing requirements as it is shown in Table I.

In the first case the IP packets are just forwarded. Hence, in this case the processor checks the several fields of the header such as the IP version, the Time to Live (TTL), the Checksum and the Destination IP and then modifies the checksum and forward the packet. In this case the Checksum module is used to check the checksum of the header, while the modification of the checksum is performed in the processor using the incremental checksum processing [28]. In the second case, the packet's header is again processed by the processor and the payload is send to the DES module for encryption or decryption. In the third case, the processor elaborates the header and the payload is sent to the IDCT module. The

payload is stored to the data RAM of the processor for further processing (i.e. transcoding).

Table I. NETWORK FLOWS

Flow	Checksum	DES	IDCT
IP Forward	✓		
Packet Encryption	✓	✓	
Packet Transcoding	✓		✓

The goal is to find the configuration of the platform with a performance that meets the demands of a specific distribution of a network workload in a given area and to make this adaptation taking into account the dynamic partial reconfiguration overhead. In order to find this configuration we analyze the architecture in the form of linear programming equations. The variable that should be optimized is the aggregate throughput of the architecture. The constraints are:

- the number of co-processors directly connected to the processor,
- the number of co-processors attached to the bus,
- the network flow requirements,
- the bandwidth of the bus,
- the available headroom of the processor, and
- the available area for the co-processors.

These requirements are described in the following equations:

$$P = \sum_{i=1}^{n} t_{ij} \cdot n_{ij} = \quad (1)$$

$$= t_{CD}n_{CD} + t_{DD}n_{DD} + t_{ID}n_{ID} + t_{CB}n_{CB} + t_{DB}n_{DB} + t_{IB}n_{IB}$$

Constraints:

$$n_{CD} + n_{DD} + n_{ID} \le MaximumNumberofDirectModules \quad (2)$$

$$n_{CB} + n_{DB} + n_{IB} \le MaximumNumberOfIBusModules \quad (3)$$

$$a_{CD}n_{CD} + a_{CB}n_{CB} + a_{DD}n_{DD} + a_{DB}n_{DB} + a_{ID}n_{ID} + a_{IB}n_{IB} \le MaximumAvailableArea \quad (4)$$

$$t_{CD}n_{CD} + t_{CB}n_{CB} \ge ForwardNetworkTraffic \quad (5)$$

$$t_{DD}n_{DD} + t_{DB}n_{DB} \ge EncryptionNetworkTraffic \quad (6)$$

$$t_{ID}n_{ID} + t_{IB}n_{IB} \ge TranscodingNetworkTraffic \quad (7)$$

$$b_{CB}n_{CB} + b_{DB}n_{DB} + b_{IB}n_{IB} \le BusMaximumBandwidth \quad (8)$$

$$c_{CD}n_{CD} + c_{DD}n_{DD} + c_{ID}n_{ID} \le \mathrm{Pr}ocessor'sHeadroom \quad (9)$$

where,
P : Aggregated throughput
t_{ij} : throughput of the *i* module connected to *j*
n_{ij} : number of the *i* modules connected using *j*
a_{ij} : area of the hardware *i* acceleration unit
b_{ij} : co-processor's bandwidth allocated in the bus
c_{ij} : number of cycles allocated in the processor
i : {C: Checksum, D: DES, I: IDCT}
j : {D: Direct, B: Bus}

The throughput of each module can be calculated using the time to send the data for processing, the time of processing and the time to receive the data. For example, in the case of the DES module connected in the OPB the time to transfer the data using DMA and the time to transfer the data to the FSL module is given by equation (10) and (11) respectively.

$$t_{DB} = (nc_{DMAInit} + words \cdot nc_{PerWordTransfer}) \cdot t_{cycle} \quad (10)$$

$$t_{DD} = words \cdot nc_{PerWordTransfer} \cdot t_{cycle} \quad (11)$$

where,
nc : number of cycles
t_{cycle} : the period of the clock

In this case, using 10ns clock cycle, the time to transfer 5 words (20 bytes) is faster using the direct connection (680ns using the shared bus and 550ns using the direct connection), while the time to transfer 16 words (64 bytes) is faster using the bus (1240ns using the shared bus and 1760ns using the direct bus). Consequently, depending on the data transfers is more efficient to use either the direct interface or the shared bus.

Besides the performance, the linear programming equation could also include power considerations. For example, the system could be solved by minimizing the power consumption of the configuration given the constraints (2)-(9) as it is shown in Equation 12 or the power consumption limits could be also added to the constraints.

$$Power = \sum_{i=1}^{n} n_{ij} \cdot power_{ij} \quad (12)$$

The percentage of the workload can be measured during the operation of the network processor. In order to make the partial reconfiguration efficient the reconfiguration overhead must be taken into account. The new configuration must be performed only if the throughput of the new configuration accumulated with the configuration overhead is greater than the current throughput as it is shown in the following equation (13).

$$Thr_{new} \cdot t \ge Thr_{old} \cdot (t + t_{reconfiguration}) \Rightarrow$$

$$t \ge \frac{Thr_{old} \cdot t_{reconfiguration}}{Thr_{new} - Thr_{old}} \quad (13)$$

where,
t : time the new configuration is active
Thr_{old} : the throughput of the previous configuration
Thr_{new} : the throughput of the new configuration

5. Implementation and Experimental Results

The design is targeting a Xilinx Virtex 2 for the MicroBlaze Framework and the Virtex 4 FX device for the PowerPC framework using the Xilinx Platform Studio v7.1.2. The DES module is provided by OpenCores [27] and the IDCT module was from Xilinx IDCT [10]. A testbench framework for IP Packet Forwarding has been developed to measure the performance of each configuration. In this testbench a packet is received and stored in the first block RAM. The processor scheduler sends the required data (header or payload) in one of the co-processors and stores the results to the Data Block RAM for further processing. The processor checks the packet's header and based on the classification it perform one of the following action:

- process the header
- process the header and encrypt/decrypt the payload
- process the header and transcode the payload.

In addition, it updates a counter that is used to measure the distribution of each network flow. When the aggregated number of packets is over a threshold (e.g. 1000 packets) the processor checks the distribution and decides to trigger a partial reconfiguration or not. After the processing by the hardware acceleration units, the packet is forwarded to the local RAM of the processor for further processing. The processor uses one block of RAM as a forward table and classifier. The IP LookUp Forward table is performed in 4 to 6 OPB accesses using the destination IP address for each access, as it is described in [11]. In the case of the MicroBlaze framework, the processor can be customized to include some additional features such as hardware multiplier, divider and barrel shifters. It is worthy to notice that the use of barrel shifter has improved the performance of the testbench program by over 30% while the area overhead is almost negligible (4 hardwired multipliers). This is due to the fact that most of the network applications include many shift instructions and the use of a barrel shifter can relieve the processor. Table II shows the throughput (in terms of packets, each packet is 512 bytes) and the area for the co-processor modules.

Table II. MODULE PROPERTIES

Module	Thoughput (Kpack./sec)	Area (slices)
Checksum-FSL	512	44
Checksum-OPB	581	87
DES-FSL	42	789
DES-OPB	54	832
IDCT-FSL	43	944
IDCT-OPB	56	987

The linear equations can be solved using any linear programming solver such as the Excel or the MPL Solver. Fig. 3 and Fig. 4 provide the optimized configuration for several workloads distribution for an average packet size of 512 bytes for the MicroBlaze and the PowerPC framework respectively. Five different workloads have been used. In the first one, 70% of the traffic is packets that need only to be forwarded, 20% belong to Virtual Private Networks (VPNs) that need either encryption or decryption, while 10% of the packets are media streams that need media processing, e.g. transcoding. The other distributions are shown in Fig. 3. The MicroBlaze framework uses 100 MHz clock frequency both for the processor, the bus and the modules. In the case of the PowerPC framework, the clock frequency of the PowerPC is 300 MHz and the remaining modules are clocked to 100MHz.

We must note that in this testbench the source of the packet is the block RAM attached to the bus and the destination is the data block RAM of each processor that further uses the processed data. Hence, in the case of the MicroBlaze, when the FSL interface is used the data are transfer from the OPB RAM to the FSL co-processors and then from the FSL co-processors to the LMB RAM. In the case that the OPB co-processor is used, the processor send the data using DMA from the OPB RAM to the OPB co-processor and after the processing the data are send to the LMB RAM. These transfers are illustrated in Table III. In the case that the packets wouldn't have to be further processed by the processor it is obvious that the DMA mechanism is much more efficient than the direct interface of the processor with the modules. Moreover, if the network module could transfer the packets to the data RAM of the processor using a dual port RAM it is obvious that the throughput of the co-processors attached directly to the processors would be more efficient; hence they would be used more often.

Table III. DATA TRANSFERS

MicroBlaze	**Source**	**Destination**
FSL modules		
	OPB BRAM	FSL module
	FSL module	LMB BRAM
OPB modules		
	OPB BRAM	OPB module
	OPB module	LMB BRAM

PowerPC	**Source**	**Destination**
APU modules		
	PLB BRAM	APU module
	APU module	OCM BRAM
PLB modules		
	PLB BRAM	PLB module
	PLB module	OCM BRAM

Fig. 3 presents the optimum configuration for several workload distributions in the case of the MicroBlaze framework. When the majority of packets need only IP forwarding, then the checksum modules are attached to both the OPB and the FSL interface. When then majority of the packets need payload processing, then the number of DES and the IDCT modules attached to the OPB bus increases and the checksum is only attached to the FSL interface. As it is shown, the payload co-processors are rarely attached to the FSL interface because of the processor's cycles that are wasted for the transfer of the data. They are only attached to the FSL interface, when there is available headroom and the maximum number of OPB slaves has been reached.

Fig. 4 presents the optimum configuration for several workload distributions for the PowerPC framework. In this framework the modules that are attached to the APU interface are less. This is mainly because the PLB bus can support burst traffics providing more throughput and using the Direct Memory Access (DMA) the processor can be off-loaded. These modules will be only attached to the direct interface only when the number of bus slaves is in the limit and there are available cycles in the processor or when the maximum number of slaves in the bus is reached. Fig. 5 and 6 shows the distribution of the modules, for the MicroBlaze and the PowerPC framework respectively, in the case of the 25/50/25 distribution for several packet sizes. As it is shown in the case of the MicroBlaze, when the average packet size is small, only a small number of payload engines is used and the checksum modules are attached to the bus. When the average size increases then the processing requirements increase. Hence more payload modules are attached to the bus, the maximum number of bus slaves is reached and the checksum modules have to use the FSL interface. In the case of the PowerPC, when the average packet size increase then the checksum modules move from the PLB bus to the APU interface. In the same figure we can see that none of the DES and IDCT modules are attached to the APU interface because of the high throughput that the PLB burst mode provides.

The main difference between the two frameworks is that in the case of the MicroBlaze the throughput is mainly constrained by the OPB Bandwidth and the headroom of the processor. On the other hand, the PowerPC framework use more efficient bus that supports burst mode and has the processing power of the processor is higher. Hence, the throughput is usually constrained by the performance of the co-processors. These figures show that the use of a reconfigurable framework that adapts its co-processors to the network distribution and the average packet size can improve the overall performance of the system.

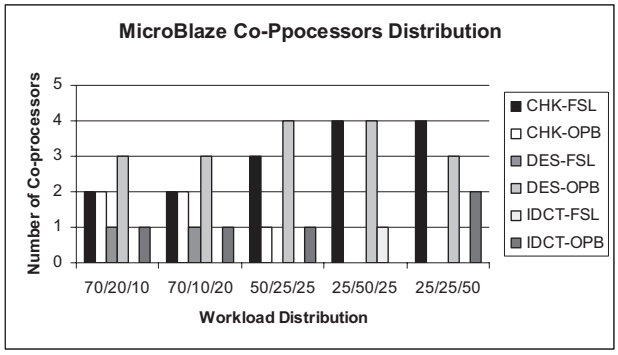

Figure 3. MicroBlaze Co-processors distribution

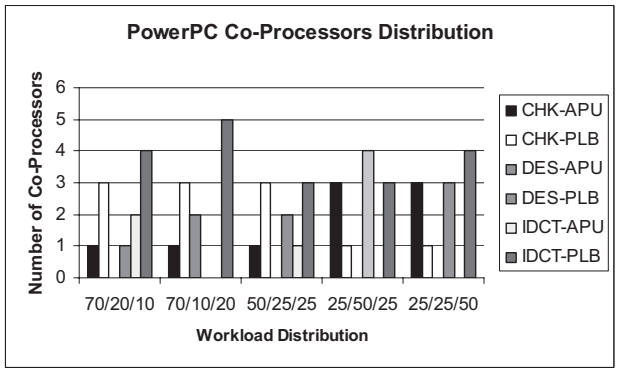

Figure 4. PowerPC Co-Processors Distribution

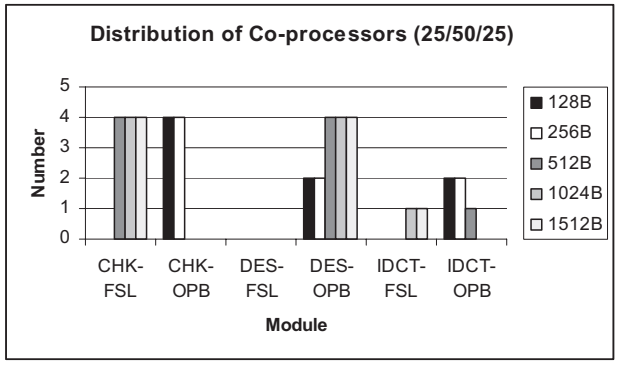

Figure 5. Distribution of modules for different packet's sizes for the MicroBlaze

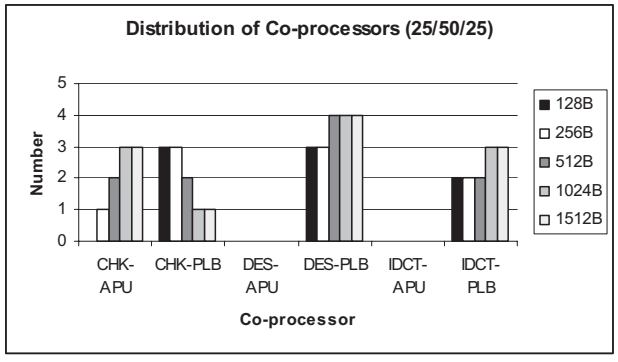

Figure 6. Distribution of modules for different packet's sizes for the PowerPC

For the implementation of the dynamic reconfigurable network processor the design has to be partitioned in reconfigurable and fixed logic modules according to the Xilinx flow for modular partial reconfiguration [16]. According to this flow, each reconfigurable area must have the height of the device; hence two reconfigurable modules were used on the left and on the right part of the FPGA. Between the fixed and the reconfigurable area special modules should be used to connect these areas, called bus macros. These modules can be the only common signals between the reconfigurable and the static area. In our case the bus macros should be used to separate the coprocessors used in the reconfigurable area with the fixed logic that contains the processors and it's RAM. In the case of the MicroBlaze framework the Bus Macros must be used in the FSL interface and the OPB bus. Fig. 7 shows the way the architecture should be mapped to the Virtex 2 FPGA platform in the case of the MicroBlaze Framework.

It must be noted that although all the possible combination of the reconfigurable area are usually too large to store into an external RAM, it is still possible to evaluate the most common network distribution and use a small RAM for storing the most efficient bitstreams for these workloads. In addition, Xilinx is towards a new design flow [17] that will enable the use of dynamically reconfigurable modules without the constraint of occupying the whole height of the device. In this case it will be possible to partially reconfigure the device by only storing the new co-processor over the previous without affecting the remaining co-processors that are attached on the same bus.

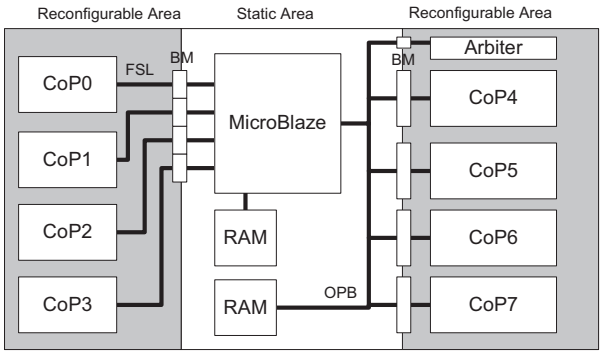

Figure 7. Floorplan of the MicroBlaze framework

Fig. 8 shows when the partial reconfiguration should be performed depending on the available area for the coprocessors based on the Equation 13 and the configuration information shown in [17]. Four different devices have been used. For each device three scenarios are calculated. In the first scenario only the FSL modules are reconfigured. In the second scenario only the shared bus modules are reconfigured and in the third scenario both the FSL modules and the shared bus modules are reconfigured for the MicroBlaze framework. As it is shown the FSL units consume less area than the OPB modules (since the FSL area is usually consists of checksum modules) hence it takes much less time to reconfigure the device.

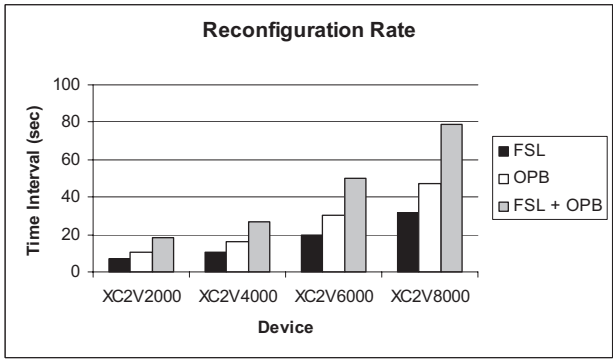

Figure 8. Reconfiguration Rate depending on the device

6. Conclusions

In this paper the design and the analysis of dynamically reconfigurable network processor is presented. The processor is embedded in an FPGA that can be dynamically partially reconfigured to change hardware acceleration units based on the distribution of network workload. The system is analyzed and formulated in order to find an optimized distribution of the co-processors that meets the network workload demands and the constraints of the platform. The experimental results show that the optimum distribution of the co-processors depends both on the average packet size and the distribution of the network. The system must be carefully designed in order to find the configuration in which both the processor and the co-processors will be fully exploited in a balanced way.

A possible extension to this work would be the development of an analytical model that can be used to explore homogeneous or heterogeneous multi-processor platforms. In this case the analysis must also include the topology of the connected processors and the ability of the co-processors and the memories to be shared by many processors.

Acknowledgement

This work was supported by Sandbridge Technologies.

References

[1] J. Carlström, T. Bodén, "Synchronous Dataflow Architecture for Network Processors", *IEEE Micro*, September-October 2004

[2] M. Venkatachalam, P. Chandra, R. Yavatkar, "A highly flexible, distributed multiprocessor architecture for network processing", *Computer Networks* , Vol. 41, pp. 563-585, 2003

[3] L. Thiele, S. Chakraborty, M. Gries, S. Künzli, "Design Space Exploration of Network Processor Architectures", *Workshop on Network Processors*, 8th International Symposium on High-Performance Computer Architecture (HPCA8), February 2002

[4] Y. Jin, N. Satish, K. Ravindran, K. Keutzer, "An Automated Exploration Framework for FPGA-based Soft Multiprocessor Systems", Proceedings of the *International Conference on Hardware/Software Codesign and System Synthesis*, September 2005

[5] T. Wolf, M. Franklin, E. Spitznagel, "Design Tradeoffs for Embedded Network Processors", Proceedings of the *International Conference on Architectures of Computing Systems* (ARCS), vol.2299, pp-146-164, April 2002

[6] H. Rosinger, "Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel", Xilinx Application Note, May 12, 2004

[7] J. Zhang, "Process XML On a Chip", CommsDesign Magazine, May 10, 2005

[8] 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

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

[10] L. Pillai, "Video Compression using DCT", Xilinx Application Note, March 3, 2005

[11] M. Degermark, A. Brodnik, S. Carlsson, S. Pink, "Small Forwarding Tables for Fast Routing Lookups", ACM *Special Interest Group on Data Communications*, SIGCOMM 1997, France

[12] K. Ravindran, N. Satish, Y. Jin, K. Keutzer, "An FPGA-Based Soft Multiprocessor System for IPv4 Packet Forwarding", Proceedings of the *International Conference on Field Programmable Logic and Applications* (FPL'05), August 2005

[13] K. Thompson, G. Miller, R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics", *IEEE Network*, vol. 11, no.6, November-December 1997

[14] S. Vassiliadis et al, "The MOLEN Polymorphic Processor", *IEEE Transactions on Computers*, pp. 1363- 1375, Vol. 53, Issue 11, November 2004

[15] S. McCreary, K. Claffy, "Trends in Wide Area IP Traffic Patterns", Technical Report from Cooperative Association for Internet Data Analysis.

[16] "Two Flows for Partial Reconfiguration: Module Based or Difference Based", Xilinx Application Note 290 v1.2, September 2004

[17] P. Sedcole, B. Blodget, J. Anderson, T. Becker, "Modular Partial Reconfiguration in Virtex FPGAs", Proceedings of the *International Conference on Field Programmable Logic and Applications* (FPL'05), August 2005

[18] K. Lahiri, A. Raghunathan, S. Dey, "System-Level Performance Analysis for Designing On-Chip Communication Architectures", *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. 20(6), pages 768-783, June 2001

[19] "MicroBlaze Processor Reference Guide", Xilinx Documentation, May 2005

[20] "PowerPC Processor Reference Guide", Xilinx Documentation, September 2003

[21] "IBM On-Chip CoreConnect Bus Architecture", IBM Documentation 2000

[23] T. Wolf, M. Franklin, "CommBench A Telecommunications Benchmark for Network Processors", Proc. of *IEEE International Symposium on Performance Analysis of Systems and Software*, April 2000

[24] P. Crowley, M. Fiuczynski, J. Baer, B. Bershad, "Characterizing Processor Architectures for Programmable Network Interfaces", Proceedings of the *International Conference on Supercomputing*, Santa Fe, May 2000

[25] P. Paulin, C. Pilkington, E. Bensoudane, "StepNP: A System-Level Explration Platform for Network Processors", *IEEE Design & Test*, v.19 n.6, p.17-26, November 2002

[26] P. Paulin, C. Pilkington, "Application of a Multi-Processor SoC Platform to High-Speed Packet Forwarding", Proceedings of the *Design, Automation and Test in Europe Conference*, (DATE'04), March 2004.

[27] S. McQueen, "Basic DES Crypto Core", OpenCores, www.opencores.org

[28] "Computation of the Internet Checksum via Incremental Update", Request for Comments 1624, May 1994