Microelectronics Journal ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Design and performance evaluation of an adaptive FPGA for network applications

Christoforos Kachris *, Stephan Wong, Stamatis Vassiliadis

Computer Engineering Lab, Delft University of Technology, The Netherlands

A R T I C L E I N F O

Article history:
Received 15 March 2007
Received in revised form
30 January 2008
Accepted 19 May 2008

Keywords:
Reconfigurable logic
Network processing

A B S T R A C T

This paper presents the design, implementation and performance evaluation of a coarse-grain dynamically reconfigurable FPGA platform for multi-service edge and access network devices. The platform consists of two MicroBlaze RISC processors and a number of hardware co-processors used for the processing of packet payloads (Data Encryption Standard (DES) and Lempel–Ziv Compression). The co-processors can be connected either directly to the processors or using a shared bus. The functionality of the co-processors is dynamically reconfigured to meet the requirements of the network workload. The system has been implemented on the Xilinx Virtex II Pro platform and the network traces from real passive measurements have been used for performance evaluation. The use of dynamically reconfigurable co-processors for network applications shows that the performance speedup versus a static version varies from 12% to 35% in the best case and from 10% to 15% on average, depending on the network traffic fluctuation.

1. Introduction

The increase of Internet traffic has created the need for more powerful and specialized network processors (NP) to be incorporated in the access and core network devices in order to sustain the demanding packet processing. NP are SoCs that consist of application-specific instruction processors (specialized for packet processing) and several memory (SRAM, DRAM) and network (GMAC, POS) interface units. NPs are utilized in a number of network devices such as servers, gateways, firewalls, etc. The architectures of NP vary from multi-core multi-threaded RISC architectures to dataflow architectures. In addition, each network processor targets different areas of the network such as core, edge, and/or access networks.

NP utilized at access and edge routers are used to process both the header and the payload of network packets. In case of header processing, hardware co-processors are used to relieve the processor of mainstream computations such as the Cyclic Redundancy Check and the checksum algorithm. These co-processors are used for every packet hence it is easy to specify beforehand the number of co-processors for a specific bandwidth. Unfortunately, in the case of the payload processing there are no standard processing requirements. For example, in a multi-service edge router, the payload processing requirements for each packet depends on the network flow it belongs to. Each network flow can have its own processing requirements such as encryption, compression, intrusion detection, intrusion prevention, etc. Since the NP target a wide area of applications, it is very difficult to design a network processor that meets all the network processing demands. In many cases, the on-chip co-processors are not able to process the required number of packets; therefore, external co-processors (for encryption, compression, or intrusion detection) are used which increase the cost and the power consumption of the device, while other on-chip co-processors are not utilized wasting valuable chip area. In addition, new protocols are certain to be deployed in the future and the NP must be able to face the new requirements. Hence, the use of reconfigurable platforms able to adapt themselves to the network processing demands is deemed necessary. Furthermore, the performance of reconfigurable logic-based systems in many network applications such as encryption [1], compression [2], or network intrusion [3] can sustain the demanding requirements of the payload processing.

The use of a dynamically reconfigurable platform for network processing can additionally be used to reduce the power consumption of a network device. The power consumption of the current NP is extremely high mainly because of the vast amount of processors running at high frequencies. The use of hardware accelerators in reconfigurable platforms can significantly reduce the power consumption, since only the necessary accelerators are configured when needed. Furthermore, as the static power of the chips starts to hold a significant portion of the

* Corresponding author. Tel.: +31 15 27 83591; fax: +31 15 27 84898.
E-mail addresses: kachris@ce.et.tudelft.nl (C. Kachris), stephan@ce.et.tudelft.nl (S. Wong), stamatis@ce.et.tudelft.nl (S. Vassiliadis).

0026-2692/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2008.05.011

overall power consumption, the use of smaller devices that can load and un-load co-processors based on the network fluctuation can further reduce the total power consumption.

Moreover, the behavior of network traffic over time is not constant in terms of packet size, network protocol, and bandwidth. In [4,5], research about how the network traffic changes over time, in terms of packet size and the packet protocols is presented. Hence, even when the NP are designed for specific applications, it is very difficult to design a processor that will always meet the network traffic demands. Therefore, the use of a dynamically reconfigurable network platform that can change the number and the type of co-processors depending on the network traffic could improve the performance of the system. In this paper, we present a case study for a multi-service edge/access router. The router must be able to support Virtual Private Networks (VPNs). In this case the packets need to be encrypted and decrypted when they are received and transmitted, respectively. In addition, many wireless devices that are attached to the router need packet compression [17] to reduce the overall bandwidth. The number of packets that need encryption and compression changes over time; hence, a network device that could be adapted to these requirements could boost the performance of the system.

The main contributions of this paper are:

- The design of a reconfigurable platform for network applications and the mapping to a Virtex II Pro device.
- The investigation of dynamic reconfiguration to improve the throughput of a network platform using co-processors connected either directly to the processors or using a shared bus.
- The performance evaluation in an application scenario with three flows (IP forward, encryption, and compression) for variable network traffic loads and several reconfiguration rates using real network traces.
- The configuration manager that controls the device based on the network stability and the packet distribution.

The paper is organized as follows. Section 2 presents the related research in the area of reconfigurable network platforms. Section 3 presents the system architecture of the network platform and Section 4 describes the implementation on a Virtex II Pro device. The analysis and the results of this design are presented in Section 5. Finally, Section 6 presents the conclusions of this work.

2. Related work

In literature, several schemes have been proposed to face the network processing requirements utilizing reconfigurable logic. This section presents the research in dynamically reconfigurable systems in the domain of network applications. In [7,8], 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 plug-in 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 plug-ins. These plug-ins are dynamically configured kernel modules used to process the active packets, that can be downloaded by a trusted server. The system has been implemented onto two FPGAs, one used as the network interface device and the other used as the host of the hardware plug-ins.

In [9], a reconfigurable system called Programmable Protocol Processing Pipeline has been introduced. In this case, a set of FPGAs is used in a pipeline manner in order to accelerate packet processing. Each 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, Forward Error Correction is used as a protocol processing function. Although FPGAs can be reconfigured dynamically, in this paper only the performance evaluation of a static design (and not of a dynamically reconfigurable device) is presented.

In [10], a reconfigurable network co-processor platform is presented called DynaCore. In this paper an FPGA-based platform is presented that can accommodate hardware accelerator units. The platform includes a dispatcher that is used to forward the incoming packets to the hardware acceleration units. The system consists only of hardware acceleration units without presenting a connection of the hardware units with the general-purpose processing elements used for the remaining header processing.

In [11], a secured adaptive network processor is presented. The use of the secured adaptive network processor both as a secure network edge device and as a user-adaptable network gateway is presented. The main characteristic of this system is that a secured mechanism is used to load the new configuration of the system, to face a possible attack on the device. The designed system was able to resist several attacks such as bus monitoring, power analysis, and timing analysis. The whole system consists of three district devices: the first device is used to perform basic packet processing, the second used for the authentication support and the configuration control, and the last used as the run-time reconfigurable device that can be used by the user to load the required functions.

In [12,13], a reconfigurable platform is introduced targeting mainly active networks. This system consists of software and hardware parts. The software part is a set of kernel and user space modules running on a Linux PC. The hardware part consists of an FPGA device that is used to load the required processing modules. When a packet is received, it is checked whether it is a passive or an active packet. For active packets the system checks whether the required hardware for this application is already present in the device. Otherwise, it can request the bitstream for the specific active packet. When a new bitstream is received for an active packet, the bitstream is authenticated, decrypted and checked for integrity and then is used for the configuration of the device.

In [14], the design and analysis of a network processor using accelerators in reconfigurable logic is presented. In this paper, two different approaches are presented. In the first case, each task is mapped to a general-purpose accelerator. In the second case, different accelerators are used for different tasks that can be dynamically reconfigured on the device. The paper showed that the use of reconfigurable modules can improve the execution time by about 20 times. The system has been evaluated in three applications: tree lookup, pattern matching, and network intrusion detection. In [24], the PLATO platform is presented. PLATO is a reconfigurable active network platform which provides four physical connections for ATM networks. Two applications were ported to this platform: an Active 4×4 ATM Switch and Wormhole IP over ATM routing filter.

The system that we present is a single-chip adaptive platform for network applications using mainstream interfaces to connect the processor with the configurable hardware acceleration units. The main characteristic is that each hardware acceleration unit is connected either to a common bus or to a direct link with the processor. Furthermore, the units can be partially dynamically reconfigured without affecting the operation of the remaining system using a configuration manager that takes into account the reconfiguration overhead. Finally, the performance of the system using real traffic traces with variable stability is presented.

3. System organization

This section presents the organization of the system that is targeting the Xilinx FPGA platform. As depicted in Fig. 1, the system consists of two 32-bit MicroBlaze soft-core RISC processors and a number of hardware acceleration units. The hardware acceleration units can be connected either directly to the processors using the Fast Simplex Link (FSL) or using a shared bus called On-Chip Peripheral Bus (OPB) that is part of the IBM Core-Connect bus [15]. The FSL is a 32-bit uni-directional point-to-point communication channel bus used to perform fast communication between the MicroBlaze and an acceleration unit. OPB is a 32-bit shared-access bus that is used to interconnect up to 16 units and supports several features such as burst modes, priorities, etc. The MicroBlaze uses one of the FSL link to communicate with a simple packet dispatcher. The MicroBlaze sends a simple command to the dispatcher and the dispatcher sends the header of the packet to the MicroBlaze. The MicroBlaze processes these headers and depending on the network flow that it belongs to, it can either simply forward it, or send it for encryption or compression before it is forwarded, as is depicted in Fig. 2. Attached to the OPB bus there is a 64 KB memory block that is used as an IP LookUp that stores the information for forwarding and classification. The algorithm that is used for finding the longest prefix match in the LookUp is the Patricia-trie algorithm used in the MiBench benchmark [16]. Patricia trie is a special form of a tree data structure that is used to store a set of strings. In contrast with a regular trie, the edges of a Patricia trie are labeled with sequences of characters rather than with single characters.

In the current design, two types of hardware acceleration units are used. The first one is the Data Encryption Standard (DES) unit for encryption that is used by the IPSec standard [17] typically in VPNs [6] and the second one is the Lempel–Ziv Compression unit that is widely used for communication with wireless devices in order to reduce the size of the transmitted packets. The DES unit is based on the OpenCores [18] modified to be attached to the OPB and the FSL interface, while the LZ compression unit is a proprietary unit. The system is divided into two parts: the static part and the reconfigurable part. The static part contains the MicroBlaze, the network interface units, the packet dispatcher, the block RAMs, one Direct Memory Access (DMA) unit, one compression unit, and one encryption unit. The reconfigurable part comprises two spare hardware units attached to the OPB bus and two hardware units attached to the FSL link (one for each MicroBlaze).

Each spare unit can be configured either as an encryption/decryption unit or as a compression/decompression unit, depending on the network traffic. If the majority of the packets belong to secured connections then the spare units can be configured as encryption units. If the majority of the packets belong to wireless connections that need compression, then the spare units can be configured as compression units. Each MicroBlaze processes the header of the incoming packet and depending on the network flow that it belongs to, it tries to allocate a resource to process the payload of the packet. The status of the configuration (which spare area contains what co-processor) is stored in a special address in a shared memory attached to the OPB bus; hence it is accessible from the MicroBlazes. Each hardware unit has a specific register used to store its status. This register is set to one when the core is busy and to zero when idle. When a processor tries to allocate the unit, it first reads this register and if it is clear then the register is set by the acceleration unit on the same access (test and set) in order to achieve an atomic operation. Subsequently, using the DMA unit, it sends the payload of the data from the Block RAM buffer to the hardware unit. The DMA unit needs 4 registers to be set for every transaction: the source address, the destination address, the length of the transfer, and the control register which also initializes the transfer. After the processing of the packet by the acceleration unit, the status register is cleared by the same unit.

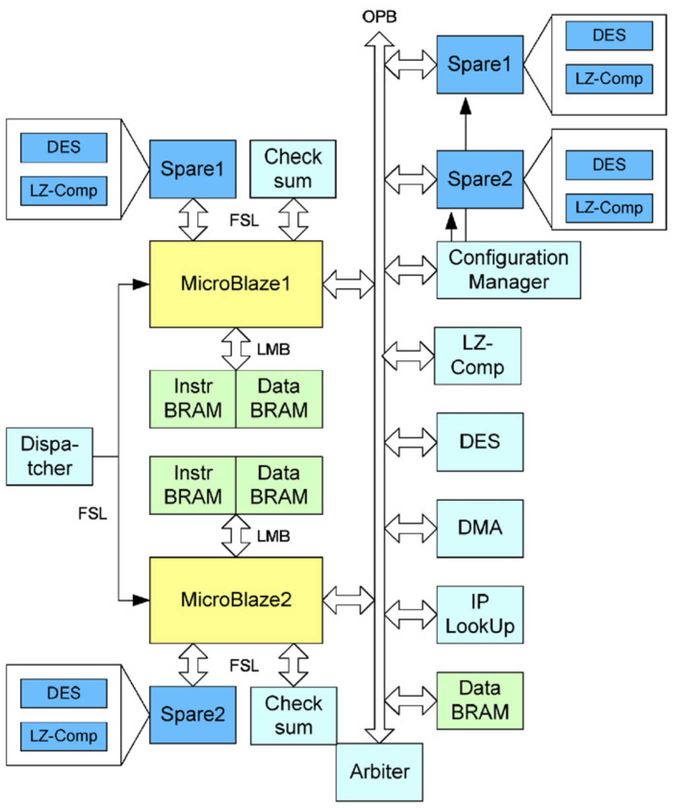

Fig. 1. System organization.

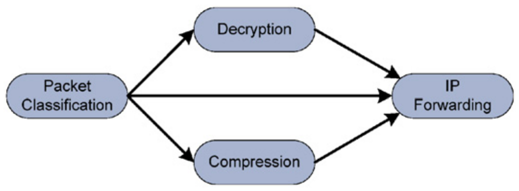

Fig. 2. Network traffic flows.

3.1. *Partial reconfiguration*

The partial reconfiguration of the device can be controlled either by one MicroBlaze or by a configuration manager. Each time a packet header is processed, the MicroBlaze updates a counter that is used for each network flow. When the total number of processed packets reaches a certain number, then the MicroBlaze checks which counter is exceeding a specific percentage of the total processed packets. In order to find the optimum sample rate (number of packets sampled) and the optimum threshold that trigger the reconfiguration, a design space exploration tool has been developed [26]. Utilizing analytical equations based on the implementation results (e.g., processing time for encrypting a packet), we can find the optimum values for sample rate and configuration threshold. This threshold is common for both of the processors and these counters are stored in a special address in the shared RAM attached to the OPB bus. Hence, the MicroBlaze that controls the configuration checks the number of processed packets belonging to each network flow for both processors.

In case that a hardware configuration manager is used, the processor can be offloaded from configuring the system. The complexity of the reconfiguration manager depends on the requirements and the fluctuation of the network traffic. A simple algorithm would be to select the configuration that is optimized for the majority of the network packets as in the case of the software configuration manager. A more advanced algorithm can also be used as depicted in Fig. 3. In this case, a Look-Up table is used in which the performance of each configuration P_{ij} is stored for several network distributions (the best configuration for each distribution is shown in the circle). The table also stores the performance of the system during the reconfiguration in which some of the co-processors are not used. At the time of the sampling, the configuration manager examines the distribution of the packets and checks if the optimum configuration for this distribution is used. In case that the network distribution has changed (hence, another configuration must be used), the configuration manager should decide if it will perform a new configuration depending on the network stability. The network stability represents the time that the distribution remains the same (within some fluctuation, e.g. $\pm 10\%$). The configuration manager should perform the reconfiguration only if the new configuration will perform better than the current configuration taking into account the reconfiguration overhead, as shown in Eq. (1). Therefore, the configuration manager will schedule a new configuration only if it will increase the overall speedup of the system. For example, Fig. 3 depicts a case in which the current configuration is C3 that is optimized for the 20/20/60 distribution (IPForward/Encryption/Compression). As the network traffic changes to 20/60/20, the configuration manager calculates whether the new configuration (C2) will speedup the system taking into account the performance degradation during the reconfiguration (Cr) and the network stability

$$P_{\text{new}} t_{\text{net\_stab-reconfig}} + P_{\text{reconfig}} t_{\text{reconfig}} \geqslant P_{\text{curr}} t_{\text{net\_stab}}. \quad (1)$$

3.2. Organization

In order to determine the number of processors and the number of hardware acceleration units for each traffic distribution, a thorough study of the requirements and the constraints of the system has to be taken into account. The constraints of the current platforms are shown in Table 1.

Table 1 shows that the system must be carefully designed to be balanced without wasting the available area. Each unit that is attached to the FSL interface of the MicroBlaze can enhance the execution of an algorithm, but many cycles are wasted for the transfer of data to and from the FSL unit, especially when the required data are loaded and stored from a RAM module attached to the OPB Bus. On the other hand, the OPB units using DMA transfers can offload the processor from demanding processing requirements, but the communication between the processor and the co-processor is slower (higher latency). In addition, the number of MicroBlaze processors is crucial to the performance of the system. An unbalanced system with many processors and reduced number of acceleration units will result in decreased performance of the system when payload processing is required and many available processor cycles will be wasted. On the other hand, the use of only one processor will result in a system with limited processing capabilities and this processor will be incapable of processing the packets and exploiting the available hardware units. In the current design, an integer non-linear programming system has been developed based on the constraints of Table 1 in order to find a balanced design [19]. Moreover, it must be noted that the MicroBlaze is a soft-core processor, in which many features such as hardware multipliers, dividers, etc. are optional. In the case of the IP forward code that is used in the current benchmark, the use of the hardware barrel shifter and the string matching unit has reduced significantly the execution time. This is due to the fact that the network functions usually include many bit-wise operations.

The system can also be configured to use the spare units of the FSL interface. But the use of FSL units has not improved the overall performance of the system. This is due to the fact that many processor cycles are wasted to transfer the data from the OPB RAM to the FSL units and back to the OPB RAM. Hence, the system increases the payload processing power but the protocol processing power is decreased. On the other hand, when that data has to be processed both by the processor and a hardware acceleration unit (such as checksum calculation or media processing) the use of acceleration units tightly attached to the processors, has shown improved performance [23].

4. Implementation

The system was mapped to a Xilinx Virtex II Pro device. Xilinx proposes two different approaches for active partial

Table 1
System's constraints

Constraint	Limit
Number of OPB units	$\leqslant 16$
Number of FSL units (per uB)	$\leqslant 8$
Bandwidth of OPB Bus	$\leqslant 500$ MB/s
Cycles for processing (per uB)	$\leqslant 100$ million
Area	$\leqslant 13\,696$ slices

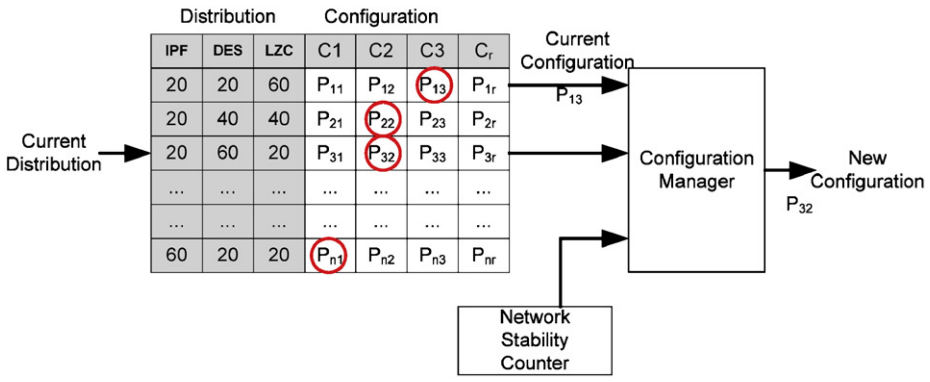

Fig. 3. Configuration manager.

reconfiguration [20]. The first one, called modular design, is used to partially reconfigure blocks of the design, while the second one, called differential reconfiguration, is used when the changes are applied only to a small number of reconfigurable elements (Look Up Tables). Our system was implemented using the modular partial reconfiguration.

According to the Xilinx design flow, in order to design a system that can be partially reconfigured, the system must be separated into static and reconfigurable areas. Reconfigurable areas must comply with specific constraints. For example, the reconfigurable module's height is always the full height of the device; the width must be a multiple of four slices, etc. In addition, the reconfigurable modules communicate with other modules by only using a special bus macro (BM). These BMs must be locked in a specific area of the device during floor planning. The only common signal of the static and the reconfigurable area is the clock signal. Xilinx provides BM that can be used to connect only adjacent reconfigurable and static areas. In [21], the design of a proprietary bus that uses BMs to add reconfigurable modules was presented. The BM can cross the static and reconfigurable areas. In our design, we created BMs that can be used for the widely used Xilinx FSL and OPB interfaces. Each OPB interface uses 108 signals; hence we need 216 signals. Each CLB row in the FPGA can be used for 4 BM wires; hence we use 54 out of the 64 available rows. These BMs have been integrated into the Xilinx Platform Studio that is used for the implementation of embedded systems.

The floor planning of the system is depicted in Fig. 4. On the left side of the device there are reconfigurable areas for the FSL interface while on the right side there are the reconfigurable OPB spare units. As depicted in the figure, the BM are the only common wires between the static and the reconfigurable areas. Table 2 shows the allocated area for each unit of the system. The area for the spare units stands for the allocated reserved area and not for the actual number of slices that each module occupies.

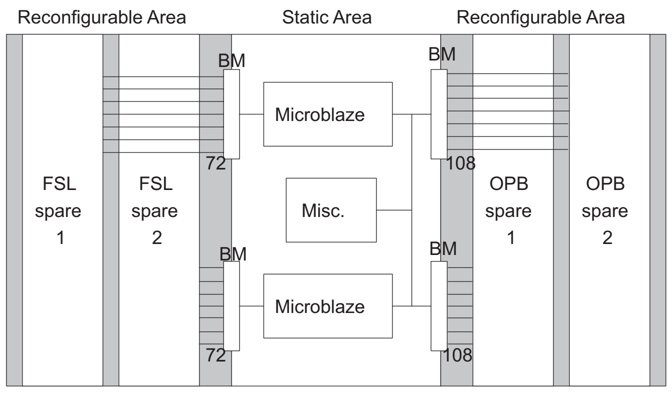

Fig. 4. System's floorplan.

Table 2
Area allocation

Block	Slices
MicroBlaze	893
DMA engine	197
OPB arbiter	180
Checksum	44
DES unit	757
LZ unit	518
FSL/OPB spare unit	1280

5. Performance evaluation

This section presents the performance evaluation of the reconfigurable system. Fig. 5 depicts the performance of the proposed system for 3 different configurations, for several processing packets sizes, and for three network distributions. In the first configuration, one OPB spare unit is used for DES encryption and one OPB spare unit is used for LZ compression. In the second configuration both of the OPB spare units are used for DES encryption and in the third configuration both of the OPB spare units are used for LZ compression. In every configuration in the static area, there is one unit for DES encryption and one unit for LZ compression. The workload distribution shows the distribution of the packets that need different processing; simple forwarding (no payload processing), encryption/decryption or compression/decompression.

As depicted in the figure, for each workload distribution there is a different configuration that maximizes the number of processed packets. When the majority of the packets need just forwarding, the balanced configuration (2DES-2LZC) has the best performance. When the majority of the packets need payload processing then the other configurations have better performance. The speedup of the dynamic configuration versus a static system with equal number of encryption and compression units varies from 12% (in the case of the 64 bytes and the 25/25/50 workload distribution) to 35% (in the case of the 64 bytes and the 25/50/25 workload distribution).

Fig. 6 presents the utilization of the OPB shared bus. When the average packet size is small (64 bytes), the utilization is small and the bottleneck of the design is the protocol processing performed by the processor (IP lookup, classification, etc.). When the average size is 256 bytes, the bottleneck of the system is the common bus since the utilization is from 60% to 90%. On the other hand, when

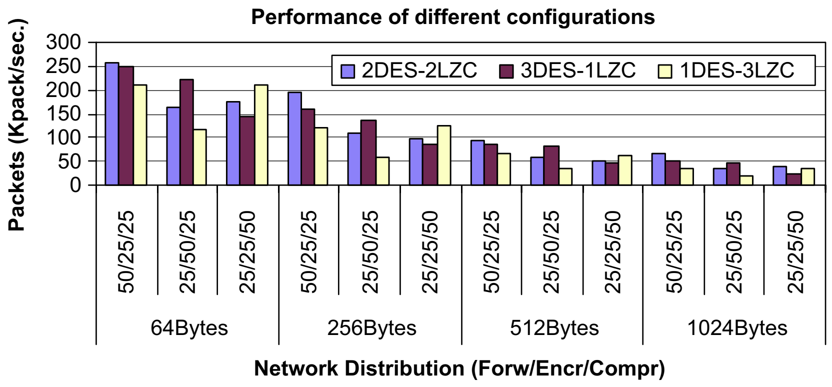

Fig. 5. Performance for different configurations.

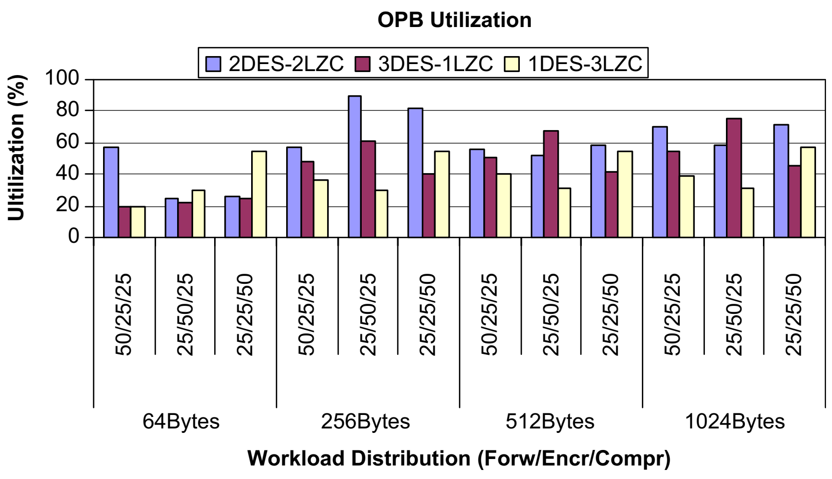

Fig. 6. OPB utilization.

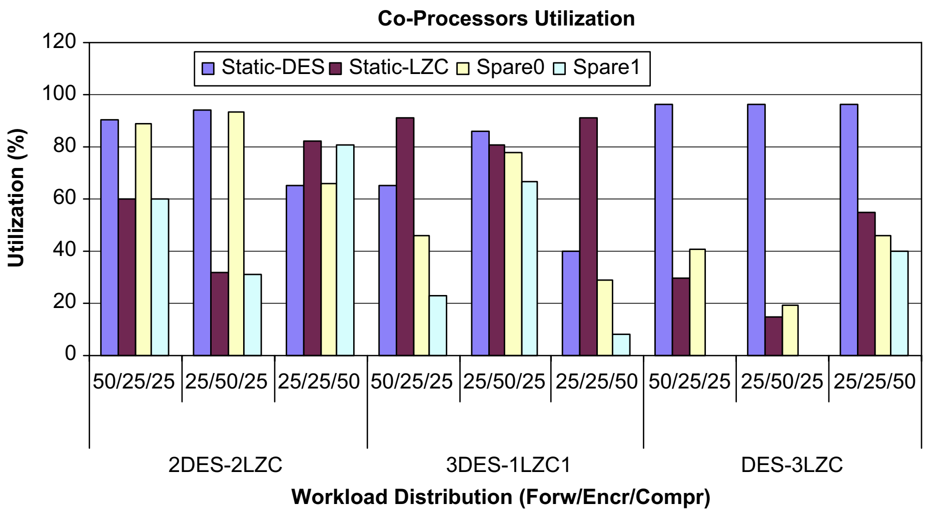

Fig. 7. Co-processor utilization.

the average size of the packets increases (512 bytes and 1024 bytes), then the acceleration units become the bottleneck of the design while the shared bus is not fully utilized.

Fig. 7 depicts the utilization of the co-processors for the three configurations and for several workload distributions. The first two columns show the utilization of the static co-processors while the other two columns show the utilization of the OPB spare units. The aggregated maximum utilization is achieved in the configuration that the system performs best. For example, in the 3DES-1LZC configuration we achieve the maximum utilization of the co-processors when the majority of the packets need encryption (25/50/25). This figure justifies the use of additional hardware units to meet the network workload. Using this figure, we can also set the distribution thresholds for the dynamic reconfiguration of the system. For example, in the second configuration (3DES-1LZC) when the majority of the packets need encryption (25/50/25), all of the DES units have high utilization. On the other hand, in the third configuration (1DES-3LZC) when the majority of the packets need compression the LZC units have lower utilization. This is due to the fact that the encryption units are more powerful. Hence, the distribution threshold that triggers the system to switch to the third configuration could be higher (60% or 70% of packets need compression).

5.1. *Run-time performance evaluation*

The main problem of the partial reconfiguration is that it cannot be done instantly. Therefore, the main goal of the system is to be able to exploit the increased performance of the different configurations by hiding the configuration overhead. A main factor to the performance of the system is the minimum reconfiguration period. If the reconfiguration of the system happens too often then the wasted time of the reconfiguration will decrease the overall performance of the system even when the new configuration is more efficient than the previous one. Another major variable is the metric of the workload distribution. The workload distribution can be measured either by counting the number of packets that belong to each network flow or by counting the number of bytes that belong to each network flow. This is due to the fact that the time to process the packet that needs only header processing is independent of the size of the packet, while in the case that the payload needs also processing the execution time is dependent on the packet size.

In order to evaluate the performance of the system with real traces, we used the network traces from the National Laboratory of Network Research. Specifically, we used the traces from the passive measurements [25] for the characteristics of the packets

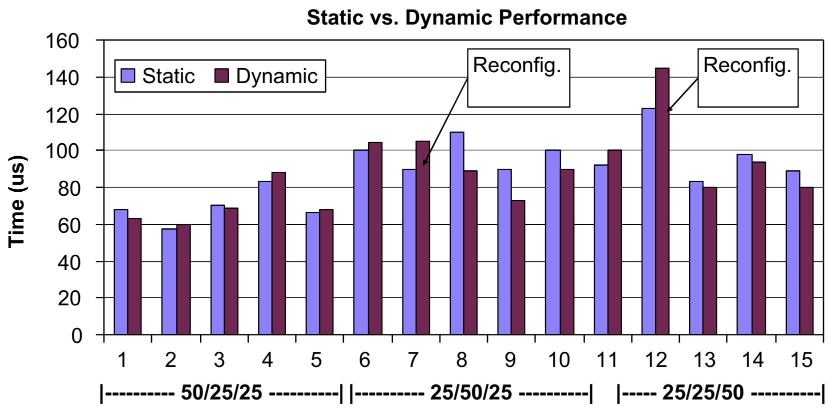

Fig. 8. Performance of the static and the dynamic system using real traces.

(size, protocol, etc.) and synthetic values for the workload distribution. We measured the performance of the system by changing the distribution of the network traffic and by changing the sampling rate of the packets. Fig. 8 shows a representative instant during this simulation in which 1500 packets are processed. Each pair of columns corresponds to 100 processed packets. In the beginning the network distribution consists of 50% of packets belonging to forwarding flows, 25% of packets belonging to encryption flows and 25% of packets belonging to compression flows. The system initialization consists of 2 encryption units and 2 compression units. After 500 packets, the packet distribution changes to 25/50/25 and after 500 packets to 25/25/50. The workload distribution is checked every 100 packets. Hence, in the instance "7" the configuration loads one more encryption unit and un-loads the compression unit. During the configuration the system consists of 2 encryption units and one compression unit, hence the time of the dynamic system slightly increases while in the instances 8, 9 and 10 that the system consists of 3 encryption units the time is decreased. During the second reconfiguration (instance 12), the two OPB spare units are reconfigured (two encryption units are un-loaded and two compression units are loaded) hence they cannot be used. Therefore, the time to process the packets increases. But during the next time periods (instances 13, 14 and 15) the total time to process the packets decreases since more compression units are used. It is obvious that the performance gain for the dynamic system is even higher when the network variation is higher (e.g., if the encryption distribution reach 80% of the network packets). Each partial reconfigurable unit uses 1280 slices and each partial reconfiguration file comprises 135 Kbytes. The reconfiguration time, according to [14,22], for the specific number of slices is almost 2.1 ms.

The speedup of the system mainly depends on the fluctuation of the network traffic. Fig. 9 depicts the speedup of the system for several sampling rates and for several network fluctuations. When the network traffic is unstable (e.g., the network flow distribution change every 100 ms) the speedup ranges from −0.4% to 12% depending on the sample rate. If the sample rate is rare (e.g., every 12 ms), then the system cannot adapt to the network traffic (most of the time the system is being reconfigured) hence there is a negative speedup compared to a static system. On the other hand, when the network traffic becomes more stable (e.g., the distribution changes every 200 ms) the speedup is always positive and the maximum speedup is performed using 5 ms sampling rate. In this case, using lower number of sample packets (e.g., 2 ms) results in lower speedup because of the small number of samples which result in false recognition of the distribution. Using higher sample rate (e.g., 12 ms) we also achieve lower speedup because of the inertia of the system to recognize the new distribution. In the case that the network traffic becomes more stable, the optimum speedup is achieved again using 5 ms sample rate. Finally, when the network traffic becomes even more stable the speedup of the system is almost the same for 5 and 12 ms sampling rate while using 2 ms packets sampling rate, the speedup is low mainly due to the false recognition of the network distribution. This figure can be used by the network system designer or the configuration manager to configure the sampling rate based on the network stability of the traffic in order to achieve the maximum speedup of the system.

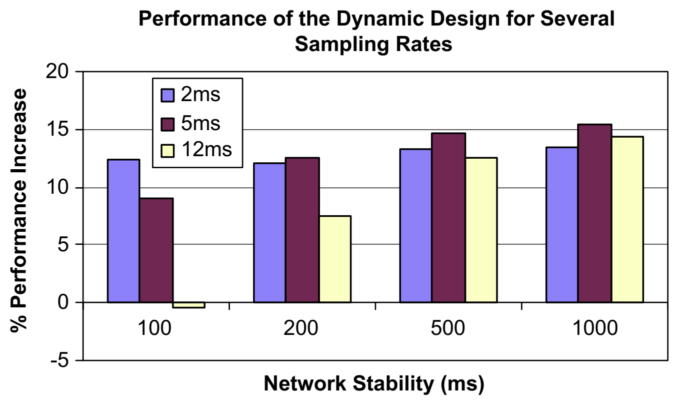

Fig. 9. Speedup versus network stability.

6. Conclusions

In this paper it is shown that the use of well-balanced dynamically reconfigurable systems can boost the overall performance of the system by 12–35% in the best case and by 10–15% on average versus a static system, as long as the network traffic changes are smooth. The configuration time, the minimum period of the reconfiguration, and the stability of the network traffic can greatly affect the performance of the system. Furthermore, the performance of the system is affected by the threshold that is used for each configuration. Therefore, the utilization of dynamically reconfigurable platforms can speedup significantly the performance of network processing system when the bottleneck of the system is the payload processing.

Acknowledgments

This work was supported by Sandbridge Technologies, Inc.

References

[1] V. Pasham, S. Trimberger, High-speed DES and triple DES encryptor/decryptor, Xilinx Application Notes, August 3, 2001.
[2] W.J. Huang, N. Saxena, E.J. McCluskey, A reliable LZ data compressor on reconfigurable coprocessors, in: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'00), Napa, CA, USA, April 2000.
[3] I. Sourdis, D. Pnevmatikatos, Pre-decoded CAMs for efficient and high-speed NIDS pattern matching, in: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'04), Napa, CA, USA, April 2004.
[4] K. Thompson, G. Miller, R. Wilder, Wide-area internet traffic patterns and characteristics, IEEE Network 11 (6) (1997).
[5] S. McCreary, K. Claffy, Trends in wide area IP traffic patterns, Technical Report from Cooperative Association for Internet Data Analysis.
[6] R. Thayer, N. Doraswamy, R. Glenn, IP security document map, Request for Comments (RFC 2411).
[7] 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), February 2001.
[8] D. Taylor, J. Turner, J. Lockwood, Dynamic hardware plugins (DHP): exploiting reconfigurable hardware for high-performance programmable routers, Computer Networks 38 (3) (2002) 295–310.
[9] I. Hadzic, W. Marcus, J. Smith, On-the-fly programmable hardware for networks, in: Proceedings of GLOBECOM98, Sydney Australia, November 1998.
[10] J. Foag, R. Koch, Architecture conception of a reconfigurable network coprocessor platform (DynaCore) for flexible task offloading, ANCHOR 2004, München, 2004, pp. 32–38.
[11] S. Harper, A secure adaptive network processor, Ph.D. Thesis, Virginia Tech, 2003.
[12] N.G. Bartzoudis, et al., Reconfigurable computing and active networks, Engineering of Reconfigurable Systems and Algorithms (2003) 280–283.
[13] N.G. Bartzoudis, et al., Active networking using programmable hardware, in: PostGraduate Networking Conference, Liverpool, June 2003.
[14] G. Memik, S.O. Memik, W.H. Mangione-Smith, Design and analysis of a layer seven network processor accelerator using reconfigurable logic, in: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'02), Napa, CA, USA, April 2002.
[15] The CoreConnect Bus Architecture, IBM Inc., September 1999, White paper.
[16] M.R. Guthaus, et al., MiBench: a free, commercially representative embedded benchmark suite, in: IEEE fourth Annual Workshop on Workload Characterization, Austin, TX, USA, December 2001.
[17] A. Shacham, et al., IP Payload Compression Protocol (IPComp), Request For Comments (RFC 2393).
[18] S. McQueen, Basic DES Crypto Core ⟨www.opencores.org⟩.
[19] C. Kachris, S. Vassiliadis, Analysis of a reconfigurable network processor, in: Reconfigurable Architecture Workshop (RAW'06), IEEE IPDPS, Rhodos, Greece, April 2006.
[20] Xilinx Inc., Two flows for partial reconfiguration: module based or difference based, Xilinx Application Notes, September 2004.
[21] J. Thorvinger, Dynamic partial reconfiguration of an FPGA for computational hardware support, Master Thesis, June 2004.
[22] P. Sedcole, B. Blodget, J. Anderson, P. Lysaght, T. Becker, Modular partial reconfiguration in Virtex FPGAs, in: Proceedings of the 2005 International Conference on Field-Programmable Logic and Applications (FPL), 2005.
[23] S. Vassiliadis, S. Wong, G. Gaydadjiev, K. Bertels, G. Kuzmanov, E.M. Panainte, The MOLEN polymorphic processor, IEEE Transactions on Computers 53 (11) (2004) 1363–1375.
[24] A. Dollas, et al., Architecture and applications of PLATO, a reconfigurable active network platform, in: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'01), Napa, CA, USA, April 2001.
[25] National Laboratory for Applied Network Research, Passive Measurements, Daily NLANR Packet Header Traces–AIX.
[26] C. Kachris, S. Vassiliadis, Design space exploration of configuration manager for network processing applications, in: IEEE International Symposium on Systems, Architectures, Modelling and Simulation (SAMOS'07), Samos, Greece, July 2007, pp. 34–40.