Performance Evaluation of an Adaptive FPGA for Network Applications

Christoforos Kachris, Stamatis Vassiliadis
Computer Engineering Lab[1]
Department of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
The Netherlands
{kachris, stamatis}@ce.et.tudelft.nl

Abstract

This paper presents the design and the performance evaluation of a coarse-grain dynamically reconfigurable platform for network applications. The platform consists of two MicroBlaze RISC processors and a number of hardware co-processors used for the processing of the packet's payload (DES encryption and Lempel-Ziv Compression). The co-processors can be connected either directly to the processors or using a shared bus. The type of the co-processors is dynamically reconfigured to meet the requirements of the network workload. The system has been implemented in the Xilinx Virtex II Pro FPGA 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 variability in time and distribution of the network traffic.

1. Introduction

The increase of Internet traffic has created the need for more powerful processors into the access and core network devices, that are able to sustain the demanding packet processing- the network processors. Network processors are specific purpose processors that can be used in a number of network applications such as servers, gateways, firewalls etc. The architectures of network processor vary from multi-core multi-threaded RISC architectures to dataflow architectures. In addition, each network processor targets different area of the network such as core networks, access networks.

The network processors are used to process both the header and the payload of the network packets. In the case of the header processing, several hardware co-processors are used to relieve the processor of mainstream computation such as the CRC (Cyclic Redundancy Check) and the checksum algorithm. These co-processors are used in every packet hence it is easy to specify beforehand the number of co-processor for the required bandwidth. Unfortunately, in the case of the payload processing there are not standard processing requirements. The payload processing requirements of each packet depends on the network flow that it belongs. Each network flow can has its own processing requirements such as encryption, compression, intrusion detection, intrusion prevention, etc. Since the network processors target a large 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, thus external co-processors (for encryption, compression or intrusion detection) are used which increase the cost of the device, while other on-chip co-processors are not used wasting valuable chip area. In addition, new protocols can be deployed in the future and the network processors must be able to face the new requirements. Hence, the use of reconfigurable platforms able to adapt themselves to the network processing demands in 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.

Moreover, the behavior of network traffic is not constant in terms of packet's size, network protocol and bandwidth over time. In [4, 5] there is a research about how the network traffic changes over time, in terms of packet's size and the packet's protocol. Hence, even when the network processors are designed for specific applications, it is very difficult to design a processor that will always meet the network traffic demands. Thus, the use of a dynamically reconfigurable network platform that can reconfigure the number and the type of co-processors depending on the network traffic, could improve the performance of the system. In a potential application scenario, a network device is used to connect at least two separated local area networks that use Virtual Private Networks (VPN). In this case the packets need to be encrypted and decrypted when they are received and transmit, respectively. In addition, any wireless device that it is attached to the network needs packets compression [17]. The number of packets that need

[1] This work was supported by Sandbridge Technologies, Inc.

Proceedings of the Seventeenth IEEE International Workshop on Rapid System Prototyping (RSP'06)
0-7695-2580-6/06 $20.00 © 2006 IEEE

encryption and compression varies over time; hence a network device that could be adapted to these requirements could boost the performance of the system.
The main contribution of this paper is:

- The design of a reconfigurable platform for network applications and the mapping to Virtex II Pro device
- The investigation of the 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 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 the implementation in 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

The use of dynamically reconfigurable systems to improve the performance of the system in many applications has been increased the last years. 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-in's 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 [9], 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 [10] 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.

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 attach on the device. The designed system was able to resist to several attacks such as bus monitoring, power analysis, and timing analysis. The whole system consists of three district devices; one device is used to perform basic packet processing, one device is used for the authentication support and the configuration control and the last one is used as the run-time reconfigurable device that can by used by the user to load the required functions.

In [12, 13] another reconfigurable platform is introduced targeting mainly active networks. This system consists of the software and the hardware part. 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 if it is a passive or an active packets. If it is an active packet the system checks if the required hardware for this application is already 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 bit-stream 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 accelerator in reconfigurable logic is presented. In this paper two different approaches are presented. In the first case each task in mapped to a general purpose accelerator, while in the second case different accelerator are used for different task that can be dynamically reconfigured in the device. As it is shown the use of reconfigurable modules can improve the execution time about 20-times. The system has been evaluated in three applications; tree lookup, pattern matching and a network intrusion detection algorithm. In [24], the PLATO platform is presented. PLATO is a reconfigurable active network platform which provides four physical connections for ATM networks. Two applications are ported in this platform; an Active 4x4 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 in a common bus or in a direct link with the processor that can be partially dynamically reconfigured without affecting the operation of the remaining system. Furthermore, the performance of the system depending on 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 it is shown in Figure 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 FSL (Fast Simplex Link) or using a shared bus called OPB (On-Chip Peripheral Bus) that is part of the IBM Core-Connect bus [15]. 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 first 40 bytes of the packet. These 40 bytes contain the Internet and the Transport's layer header when these headers do not have any option fields (the most common case). In case that the processor needs more data of the packet it sends one more command, and receives the next 40 bytes. The MicroBlaze processes these headers and depending on the network flow that it belongs, it can either simply forward it, or send it for encryption or send it for compression. Attached to the OPB bus there is 64Kbytes memory block that is used as an IP LookUp that stores the information for the forwarding and the classification. The algorithm that is used for searching for the longest prefix match in the LookUp is the Patricia-trie algorithm used in the MiBench benchmark [16].

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 typically in Virtual Private Networks (VPN) [6] and the second one is the Lempel-Ziv Compression unit that is used in IPSec standard as it is described in [17] and is widely used to connect wireless devices. 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 contains two spare hardware units attached to the OPB bus and two hardware units attached to the FSL link (one for each MicroBlaze).

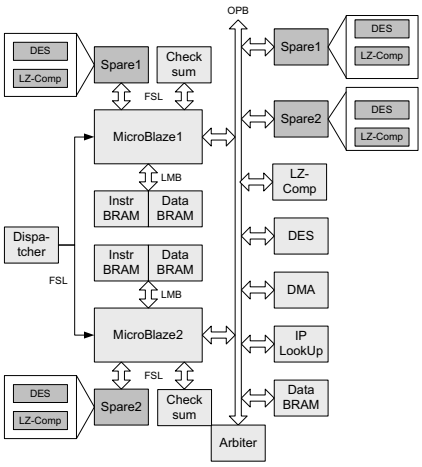

Figure 1. System Organization

Each spare unit can be configured to contain either the encryption/decryption unit or the compression/de-compression unit, depending on the network workload. Each MicroBlaze process the header of the processing packet and depending on the network flow that it belong, 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 that is used to store the status of this unit. When the core is busy the resister is set and when the core is free it is zero. When a processor try to allocate the unit, it first reads this register and if it is clear then it sets the register on the same access unit in order to prevent deadlocks (test and set). Then, 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.

The partially reconfiguration of the device is controlled by one MicroBlaze. Each time a packet's header is processed, the MicroBlaze updates a counter that is used for each network flow. When the total number of packet is over a threshold the MicroBlaze checks which counter is over 40% of the total processed packets. 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 of the processors.

In order to determine the number of processors and the number of hardware acceleration units a thorough study of the requirements and the constraints of the system have been taken into account. The constraints for the current platforms are shown in Table I:

Table I. System's Constraints

Constraint		Limit
Number of OPB units	≤	16
Number of FSL units (per uB)	≤	8
Bandwidth of OPB Bus	≤	500 MB/s
Cycles for processing (per mB)	≤	100 million
Area	≤	13696 slices

This table shows that the system must be carefully designed to be balanced without wasting available area. Each unit that it is attached to the FSL interface of the MicroBlaze can enhance the execution of an algorithm, but a lot of cycles are wasted for the transfer of the 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. In addition, the number of MicroBlaze's is very important to the performance of the system. An unbalanced system with many processors and reduced number of acceleration unit will result to 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 to a system that the processing capabilities of this processor are not enough to process the packet and exploit the available hardware units. In the current design, an integer non-linear programming system has been developed based on the constraint of Table 1 in order to find a balanced design [19]. Moreover, it must be noted that the MicroBlaze is a soft core processors, in which many features such as hardware multipliers, dividers etc. can be added to the processor or not. 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 the time of execution over 30%. This is due to the fact that the network functions usually include many bit-wise operations.

4. Implementation

The system has been mapped to a Xilinx Virtex II Pro XC2VP30 device. Xilinx proposes two different approaches for active partial reconfiguration [20]. The first one, called modular design, is used to partial reconfigure blocks of the design, while the second one, called differential reconfiguration, is used when the changed are applied only to a small number of reconfigurable elements (Look Up Tables). Our system has been 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 underlie into 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. These bus macros (BM) must be locked in a specific area of the device during the floor-planning. The only common signal of the static and the reconfigurable area is the clock signal. Xilinx provides bus macro that can be used to connect only adjacent reconfigurable and static areas. In [21] it has been presented the design of a proprietary bus that uses bus macros to add reconfigurable modules. The bus macros can cross the static and reconfigurable areas. In our design we have created bus macros 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 bus macro wires; hence we use 54 out of the 64 available rows. (The Virtex II Pro 30 device has 80 rows but 24 of them are allocated to the PowerPC area). These bus macros have been integrated into the Xilinx Platform Studio that it used for the implementation of embedded systems.

The floor-planning of the system is depicted in Figure 2. In the left side of the device there are the reconfigurable areas for the FSL interface while in the right side of the device there are the reconfigurable OPB spare units. As it is shown, the bus macros 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.

Table II. Area allocation

Block	Slices
MicroBlaze	893
DMA Engine	197
OPB Arbiter	180
Checksum	44
DES Unit	757
LZ Unit	518
FSL Spare Unit	1280
FSL Spare Unit	1280

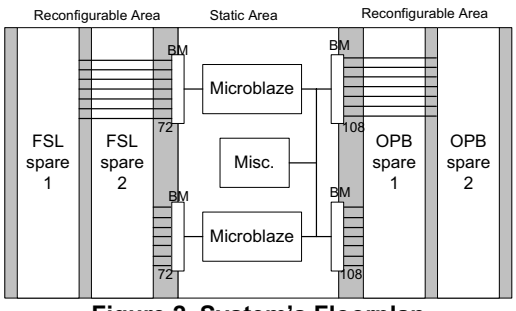

Figure 2. System's Floorplan

5. Performance Evaluation

This section presents the performance evaluation of the reconfigurable system. Figure 3 present the impact of the configuration in the performance of the system for four different packet's sizes; 64, 256, 512 and 1024 bytes. As it is shown, we use three configurations. 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/de-compression.

As it is shown, 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).

Figure 4 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, we have a balanced system between the processor, the co-processors and the interconnection (bus) hence we have the maximum utilization. On the other hand, when 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.

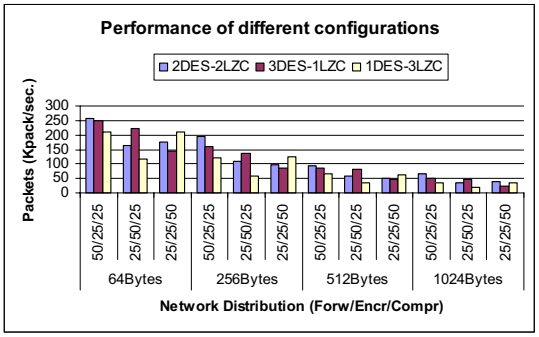

Figure 3. Performance for different configurations

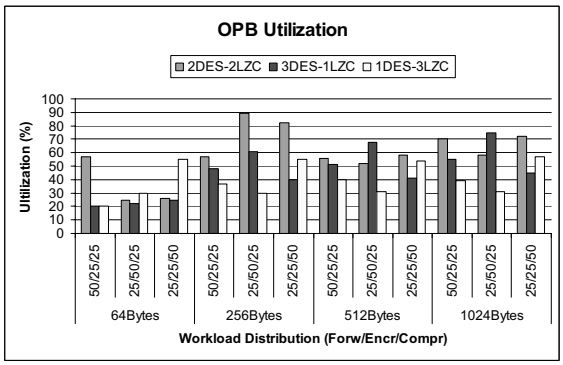

Figure 4. OPB utilization

Figure 5 presents 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. As it is shown, 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).

The main problem of the partial reconfiguration is that it can not be done instantly. Hence, 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 need 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's size.

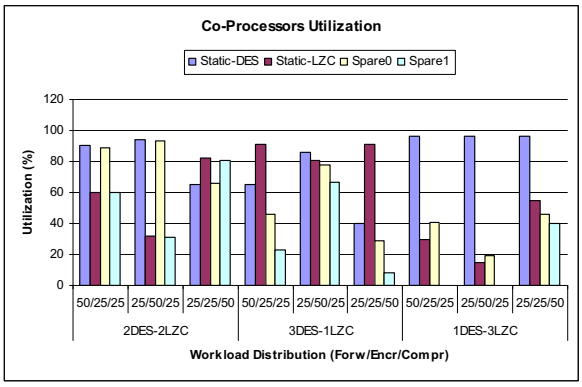

Figure 5. Coprocessor utilization

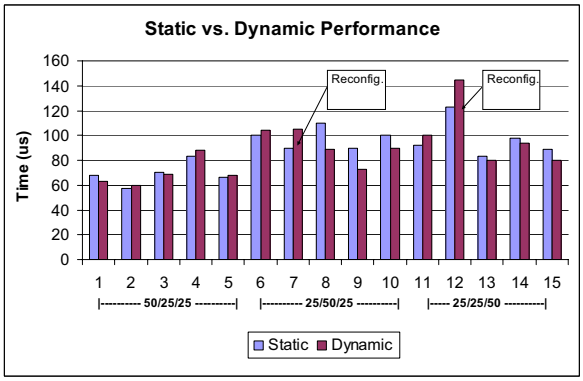

Figure 6. Performance of the Static and the Dynamic System with real traces

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 (size, protocol, etc.) and synthetic values for the workload distribution. We measured the performance of the system by changing the distribution of the network and by changing the sampling rate of the packets. Figure 6 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 more 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 increase 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 can not be used. Thus the time to process the packets increase. But during the next instances, (instances 13, 14 and 15) the total time to process the packets decrease 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 is 1280 slices and each partial reconfiguration file is 135 Kbytes. The reconfiguration time, according to [14, 22], for the specific number of slices is almost 2.1ms (66Mbytes/sec).

Fig. 7 shows the speedup of the dynamic platform over the static version for several sampling rates and network variations. In every case, the traffic of the network is equally divided in the 3 cases of network distribution (50/25/25, 25/50/25 and 25/25/50). In the first case the distribution of the packets remain the same for 100ms. When the system checks the distribution of the packets (sampling rate) every 2ms then the dynamic system is 12.4% better than the static system. When the sampling rate is rare (12ms) then the system can not follow the changes of the network traffic and keeps reconfiguring the system; hence the performance is worse than the static version. When the network traffic becomes more stable (e.g. the network distribution remains the same for 500 or 1000ms) then the performance of the dynamic system is almost 15% better than the static system.

In this case, when the sampling rate is 2ms, the overhead makes this configuration slightly worse than using 10ms sampling rate. On the other hand when the sampling rate is 12ms the system can not follow the changes in the network; hence the speedup is also slightly worse than the 10ms sampling rate. Thus, in this case the sampling rate of 10ms gives the best performance. It is obvious that if the variability of the distribution was larger (e.g. the encryption traffic could be 70 or 80% of the network traffic) the speedup of the dynamic systems over the static system would be even more. This figures, shows that the sampling rate of a dynamic platform for network applications should be carefully defined based on the variability of the network distribution and the network traffic.

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 the many processor cycles are wasted to transfer the data from the OPB RAM to the FSL units and back, while the processor does not actually processes these data. Hence, the system increase the payload processing power but the protocol processing power is decreased.

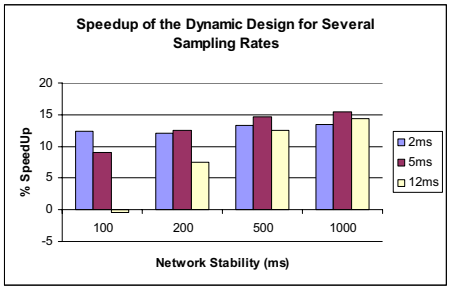

Figure 7. Speed Up of the dynamic system

On the other hand, when that data have 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 processor, has shown improved performance [9].

6. Conclusions

As it is shown in this paper the use of well balanced dynamically reconfigurable systems can boost the overall performance of the system by 12% to 35% on the best case and by 10% to 15% on the 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 it is used for each configuration. Hence, the exploitation of dynamically reconfigurable can speedup the performance of network processing system when the performance is mainly depends on the payload processing requirements of the system.

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", *IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM'00), April 2000, Napa CA, USA

[3] I. Sourdis, D. Pnevmatikatos, "Pre-decoded CAMs for Efficient and High-Speed NIDS Pattern Matching", *IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM'04), April 2004, Napa CA

[4] K. Thompson, G. Miller, R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics", *IEEE Network*, vol. 11, no.6, November-December 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)", 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*, vol. 38, no. 3, pp. 295-310, February 2002

[9] I. Hadzic, W. Marcus, and J. Smith, "On-the-fly Programmable Hardware for Networks", in Proceedings of *GLOBECOM98*, Sydney Australia, November 1998

[10] Foag, J., Koch, R., Architecture Conception of a Reconfigurable Network Coprocessor Platform (DynaCore) for Flexible Task Offloading. *ANCHOR 2004*, 32-38, München 2004

[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", *PostGraduate Networking Conference*, June 2003, Liverpool.

[14] G. Memik, S. O. Memik, W. H. Mangione-Smith, "Design and Analysis of a Layer Seven Network Processor Accelerator Using Reconfigurable Logic", *IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM'02), April 2002, Napa CA

[15] "The CoreConnect Bus Architecture", IBM Inc., Sept. 1999, White paper

[16] Matthew R. Guthaus et al., "MiBench: A free, commercially representative embedded benchmark suite", *IEEE 4th Annual Workshop on Workload Characterization*, Austin, TX, 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", *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*, pp. 1363-1375, Vol. 53, Issue 11, November 2004

[24] A. Dollas *et al*., "Architecture and Applications of PLATO, a Reconfigurable Active Network Platform", *IEEE Symposium on Field-Programmable Custom Computing Machines* (FCCM'01), April 2001, Napa CA

[25] National Laboratory For Applied Network Research, Passive Measurements, "Daily NLANR Packet Header Traces -AIX"