# Design Space Exploration of Configuration Manager for Network Processing Applications

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

***Abstract*—Current FPGAs provide a powerful platform for network processing applications. The main challenge is the exploitation of the reconfiguration to increase the performance of the system. In this paper, a design space exploration framework is presented to design a reconfigurable platform for multi-service network processing applications. An integrated design flow is presented from the system level analytical design to the implementation level. Furthermore, the design of an efficient configuration manager is presented in which the platform adaptation is performed for optimum speedup with minimum overhead taking into account the reconfiguration overhead and the network characteristics (packet type distribution, network stability). Finally, a case study is presented in which the platform is used to process three network flows with different processing requirements.**

## I. INTRODUCTION

The increase of network traffic and the emerging network applications have created the need for more powerful network processing devices. The first network devices were using general-purpose processors to perform simple header processing. Currently, the emerging applications such as VoIP, Video-On-Demand, Virtual Private Networks (VPNs), wireless networks, and others require not only simple header processing but also powerful payload processing (e.g., encryption, compression, and intrusion detection). These requirements are mainly faced at the edges of the networks, while in the core networks the majority of the packets need simple header processing. The current network processors have been evolved to multi-threaded multi-processor platforms that can face the header processing requirements. Although the majority of the network processors incorporate some hardware acceleration units, these units cannot handle efficiently the increased payload processing requirements.

Hence reconfigurable platforms (e.g., FPGAs) provide a viable and power efficient alternative for the demanding payload processing requirements. The reconfigurable platforms can be used to achieve both higher performance and lower power consumption compared to a network processor since hardware accelerators are used instead of software processing that consume less power. The main benefit of the FPGAs is that they can be reconfigured to meet the workload requirements, thus they can adapt to the fluctuations of the network traffic. Furthermore, some of the modern FPGAs can be partially reconfigured; hence the static part of the system remains operational. The main drawback of the reconfiguration is that during reconfiguration the system is partially working hence the performance is decreased compared to a static system. Furthermore, the reconfiguration scheduler must be carefully designed not to add overhead. Hence, a reconfigurable platform for network processing must be carefully designed in order to exploit the reconfiguration without adding overhead to the system. In addition, the dynamic reconfiguration can be used to lower the power consumption of the network devices. The static power consumption keeps increasing and today is a significant fraction of the total power consumption. The use of dynamic reconfiguration can result to the use of a smaller device (hence less number of gates) while providing the same performance. For example, the quiescent power of an FPGA device with 40K slices is almost the same with the dynamic power consumption of 20K slices using a 100MHz clock frequency and 12.5% toggle ratio [16]. Hence, the use of dynamic reconfiguration can result to a lower cost and lower power solution.

This paper presents an integrated framework for the design space exploration of a reconfigurable platform for network processing applications and can be used to tune the reconfiguration manager depending on the systems characteristics and the network features (network distribution, network stability, etc.). A case study is presented in which the platform has to process three network flows with different processing requirements. The system is explored in terms of network stability, reconfiguration overhead, and average packet size to determine the sample rate and the network threshold to achieve higher speed ups. The contributions of this paper are:

- An integrated framework for the design space exploration of reconfigurable platforms in network processing applications
- A methodology to find the optimum scheduler configuration in terms of sample rate and network threshold
- A case study of reconfigurable network processor that needs to process three different flows for changing network stability

Section 2 presents the related work in the area of design space exploration and design of configuration managers. Section 3 presents the reconfigurable platform for network processing and the proposed design flow. Section 4 presents a case study for a network device that processes three different flows, the design space exploration for several scenarios and the framework for the tuning of the configuration manager. Finally, section 5 presents the conclusions and the future work.

## II. RELATED WORK

The use of design space exploration (DSE) tools can be very useful when designing the micro-architecture of the network processor. EXPO [1] is a DSE tool that uses the theory of the arrival and service curves to model the operation of network processors. 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 [2] 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 for network processing applications. The CommBench [14] benchmark 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 [3] 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.

Finally, STMicroelectronics has presented a system-level exploration platform for Network processors called StepNP [4], [5]. 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. All of these frameworks are used to perform design space exploration to find the optimum static architecture for specific network traffic. In the area of reconfigurable logic, many have proposed the use of FPGAs to accelerate the performance of the system by exploiting the dynamic reconfiguration. In all of these cases the system is targeting multimedia applications in which the co-processors are scheduled based on the application task graph.

In [10], a reconfiguration manager is presented to hide the reconfiguration latency. The manager applies two different techniques at run-time: prefetch scheduling and replacement. In the prefetch scheduling technique, the manager schedules the reconfiguration based on scheduled sequence of tasks and their loading latency. Furthermore, they apply an intertask optimization technique to further decrease the reconfiguration overhead. In [11] various types of prefetching to reduce the reconfiguration overhead are also applied for the configuration manager. They present techniques such as static, dynamic and hybrid configuration prefetching. Furthermore, the configuration manager applies relocation and defragmentation techniques to reduce the dynamic reconfiguration overhead. A similar approach is presented in the MOLEN framework [13], in which a polymorphic processor is presented incorporating both general purpose and custom computing processing. The reconfiguration is mainly scheduled by the software, in which the hardware accelerators are pre-loaded in order to decrease the reconfiguration overhead.

In [15] a framework for reconfigurable computing scheduling is presented in which the main task is the scheduling of the reconfigurable units at design time. A similar approach is also presented in [8][9], in which the optimum scheduling sequence is investigated based on the task graphs of the applications. In all of these approaches the scheduler can decide based on the task graph the sequence of the reconfiguration. On the other hand, in the area of network processing the workload is dynamic hence the management of the reconfigurable units should be performed on-the-fly. Furthermore, the scheduling of the reconfigurable units should be simple enough so there is no much processing overhead. Hence, complicated algorithm should be avoided that will consume part of the processing power.

Finally, in [12] a configuration management for network devices is presented. The proposed architectural framework incorporates a mobile agent based methodology for the networked reconfigurable embedded devices. The paper presents the system level framework in which the embedded devices that are based on FPGAs can be used similar to software upgrades. Hence, the system is mainly based on the higher level of the architecture and not on the dynamic reconfiguration at run-time to adapt the system to the network. Also in [17] a case for run-time adaptation in packet processing systems is presented that targets the Intel network processors. The proposed framework is used to allocate several micro-engines (simplified processors) depending on the fluctuation of the network flow. Each micro-engine is used for separate flow, thus the system is used to automatically allocate a specific number of processor for each flow.

## III. SYSTEM DESCRIPTION

The proposed system is targeting reconfigurable platforms (e.g., FPGAs) and consists of a number of specialized processors and a number of co-processors (hardware accelerators). The block level diagram of the system is shown in Figure 1. It consists of a shared bus to which both the processors and the co-processors are attached. Moreover, additional co-processors can be attached directly to the processor using dedicated interfaces. The system can be logically divided into two parts. The first part is static while the second part can be reconfigured. The static part incorporates the processors, the shared bus, the network interface units, the memory controllers and some of the co-processors. On the other hand, the reconfigurable part

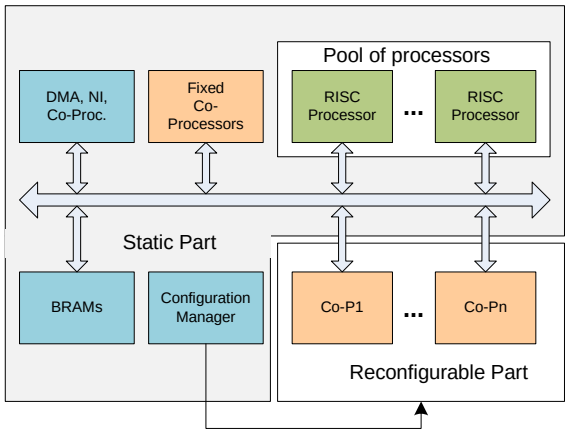


**Figure 1. System Block Diagram**

incorporates the co-processors units and it is controlled by the configuration manager. The whole system is based on the Xilinx Virtex4 FPGAs that support partial reconfiguration; hence the static part is operational while the reconfiguration part is being reconfigured.

The network processors that are used at the edge of the networks (e.g. edge routers) have to process several flows with different processing requirements. As a case study, Figure 2 shows a task graph in which the packets are classified into three different flows. In case that the packets belong to a VPN connection that payload must be encrypted. If the packets are sent to wireless devices, the payload needs to be compressed. In all other cases, the packets are just forwarded to the next network device. The main challenge of a system designer is to create a balanced system in which the network processor can face the network traffic. The main advantage of the reconfigurable platform is that it can be reconfigured to the best configuration for each network traffic distribution. Furthermore, the use of reconfigurable platforms for network processing can be used to upgrade the system with new updated accelerators or the use of new co-processors for emerging technologies (VoIP, Video-on-Demand, etc).

The main challenge in the design of a reconfigurable platform is to find the best configurations for each traffic flow distribution and then to design a configuration manager that will be tuned to exploit the reconfiguration based on the changing network traffic taking into account the reconfiguration overhead. The proposed design flow for the efficient design of a reconfigurable platform for network processing is shown in Figure 3. The first task is the system analysis, then the performance evaluation and finally the design space exploration and the tuning of the configuration manager.

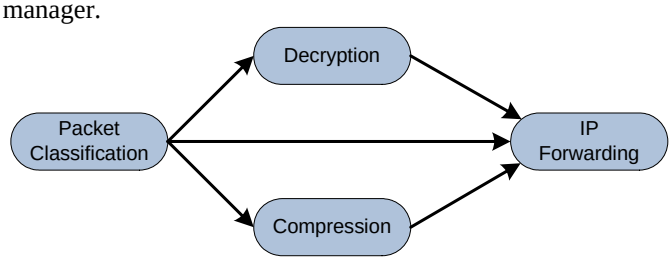


**Figure 2. Network Traffic Flows**

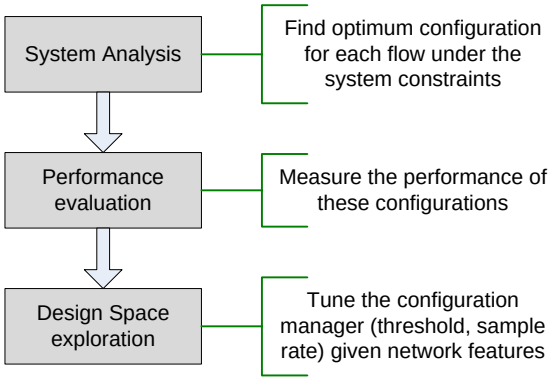


**Figure 3. Design Flow**

The system analysis is used to perform an analytical design space exploration of the platform at the system level and to find the optimum configuration for each workload. The system is described in terms of integer linear programming in order to find the best configuration in terms of number of processors, number of co-processors and type of connection with the processor [6]. The co-processors can be connected either directly with the processor using direct interfaces or can be attached to the shared bus. The direct interface provides faster interconnection between the processor and the co-processor with reduced latency. On the other hand, the co-processors attached to the shared bus provide higher throughput using the DMA burst functions but introduce higher latency. The whole system is described as an integer linear programming problem in which the processing time of a specific flow distribution is the function that must be minimized based on several constraints. The constraints of the current system are the following:

- Maximum number of units attached to the bus
- Maximum number of units attached to the processor
- Maximum bandwidth of the shared bus
- Processing power requirements
- Minimum Quality-of-Service requirements for some specific flows
- Maximum Reconfigurable and Static area

Using these constraints we can perform an analytical design space exploration and find the best configuration for each network flow distribution. The solution of these equations provide the number of processors, the number of each co-processor and the connection with the processors (direct or using the shared bus). The detailed framework is described in [6]. The results depend on the type of processor (hard-core or soft-core), the bandwidth of the bus (64-bit or 32-bit) and the average packet size. The hardware accelerators that handle header data tend to be attached directly to the processor while the co-processors that use the packet's payload tend to be attached to the shared bus. The following table shows the best configuration for each network flow distribution. For the flow distribution we used the flows shown in Figure 2. The first coprocessor is used for DES encryption/decryption and the second co-processor is used for Lemplel-Ziv Compression.

**Table 1. Configuration per Traffic Distribution**

| Network Flow Distribution (Plain-Encr.-Compr.) | Opt. Configuration (DES-LZCompr) |
|---|---|
| 50%-25%-25% | 2-2 |
| 25%-50%-25% | 3-1 |
| 25%-25%-50% | 1-3 |

After the system level design space exploration, a more detailed performance evaluation can be performed to measure the real throughput of the system for each configuration. In the performance evaluation the system is developed and simulated using real and synthetic traces to measure the performance of the system [7]. The distribution of the network flows (percentage of packets that belong to each flow) was synthetic in order to measure the performance of the system for several network flow distribution. The performance evaluation can be used to measure in more details several features of the system such as the exact processing time, the utilization of the processors and the co-processors, the utilization of the bus and the power consumption of each configuration. Furthermore, the implementation of the system can also provide information about the reconfiguration overhead. The reconfiguration time (the time to partially reconfigure the device) depends on the area of the co-processors (in terms of slices for the FPGAs) and the sequence of the configurations. The following table (Table 2) shows the execution time for each packet (assuming 256 bytes as average packet size) belonging to different flow per configuration and Table 3 shows the reconfiguration time depending on the number of reconfigurable units.

**Table 2. Processing time per configuration**

| Configuration | Processing (usec) | | |
|---|---|---|---|
| | Forward | Encrypt | Compress |
| 1. 2DES-2LZC | 0.1 | 8.4 | 7.3 |
| 2. 3DES-1LZC | 0.6 | 2.4 | 12.3 |
| 3. 1DES-3LZC | 0.6 | 16.5 | 1 |
| 4. 1DES-1LZC | 0.2 | 12.9 | 5.3 |

**Table 3. Reconfiguration time**

| Number of co-processors | Configuration Time |
|---|---|
| 1 | 2.1ms |
| 2 | 4.2ms |

Finally, a design space exploration of the system should be performed to tune the configuration manager in order to achieve the maximum speed up. The design space exploration takes into account the network stability, the reconfiguration time and the average packet size to tune the manager in terms of network threshold and sample time. The network threshold is used by the configuration manager to decide when it should perform a reconfiguration. If the sample rate is small then the configuration manager will have a major overhead to the system and the configurations will be too often. On the other hand if the sample rate is large (thousand of packets) then the system does not adapt fast enough to the network traffic hence a reduced speed up is achieved. Hence, a design space exploration tool is developed to check if the reconfiguration can achieve higher throughput than a static system and then find the configuration values for the threshold and the sample rate.

## IV. DESIGN SPACE EXPLORATION AND TUNING

After the performance evaluation of the implemented design a more accurate design space exploration can be performed. A framework has been developed in Matlab in which the following operations are performed:

- Calculation of the speedup based on
  - The network stability
  - The reconfiguration time
  - The sample rate
  - The network threshold
- Design space exploration to find the optimum sample rate
- Design space exploration to find the optimum network threshold
- Design space exploration to find the best sample rate and threshold for given network stability and reconfiguration time.

Figure 4 shows the GUI of the framework for a specific simulation in which the traffic distribution change 2 times, thus the system is reconfigured each times. The system is initialized in the balanced configuration (2 units for encryption and 2 units for compression), then switch to the encryption-optimized configuration and then to compression-optimized configuration. Each integer represents a configuration from Table 2. The state in which the reconfiguration takes place and the spare units cannot be used is symbolized as 4. As it is shown in the figure the reconfiguration time when the system change from Config2 to Config3 is double as the others, since both of the reconfigurable spare units are reconfigured. The lower figure shows the processing time per number of samples packets (in this case 100 packets). As it is shown, during the reconfiguration the processing time is much higher than the static system. This is due to the fact that the spared co-processor units that are reconfigured cannot be used. But after the reconfiguration the processing time of the dynamic system (dashed line) is lower than the static system. Hence, this framework is used first to explore the speedup of the reconfigurable system for several network workloads. The network stability represents the time in which the distribution of the flows remain the same (e.g. 50% of the packets need just forwarding, 25% need compression and 25% need encryption) with some variation (e.g. ±5%). If the network stability change too often then the system is reconfigured too often hence a negative speedup is achieved due to the reconfiguration overhead. The network stability values can be measured from the targeted network. In order to have a positive speedup the performance of the system with the new configuration and during the reconfiguration time must be better than the performance of the system of keeping the previous configuration as it is shown in the following equation:

$$P_{new}t_{new} + P_{config}t_{config} \geq P_{old} \cdot (t_{new} + t_{config}) \quad (1)$$

where,

$t_{new}$ : the time that the new configuration is active
$t_{config}$ : the time for the reconfiguration
$P_{new}$ : the performance of the new configuration
$P_{reconfig}$ : the performance during the configuration
$P_{old}$ : the performance of the previous configuration

Figure 5 shows how the speedup of the reconfigurable system depends on the network stability and the reconfiguration time. When the network distribution changes too often (5ms) then the speedup is usually negative. As the network become more stable (the distribution remain the same for more than 10 ms), the speedup increase up to the maximum speedup (the maximum speedup is calculated assuming a zero reconfiguration time). In the last case that the network is quite stable (30ms) the reconfiguration time has small effect on the speedup of the system. The reconfiguration time depends on the number of co-processors that are reconfigured and the features of the FPGA device (height and width in terms of slices).

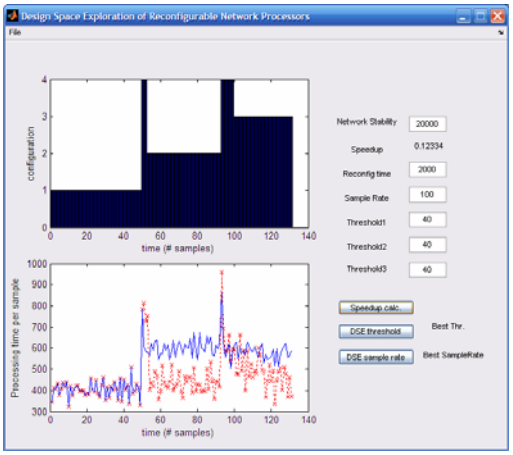


**Figure 4. Speedup Calculation**

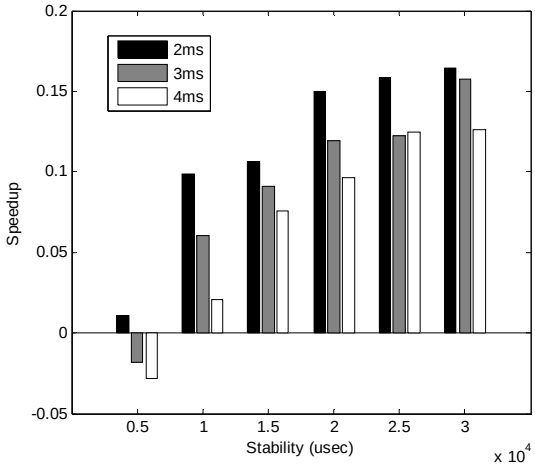


**Figure 5. Speedup vs. network stability**

Furthermore, the framework can be used to specify the network threshold in which the system must be reconfigured. If the network threshold is too low then the reconfiguration will happen too often. On the other hand if the threshold is too high then the system will fail to adapt to the workload traffic. The design space exploration can be achieved using any heuristic algorithm. But since the possible values is bounded (0-100), a two-phase exhaustive search is performed. In the first phase the percentage is increased every 10% and then a more detailed second phase is performed around the maximum value of the first-pass. As it is Figure 6, the best threshold for the change of the reconfiguration from the balanced configuration to the encryption-optimized configuration is around 40% of encrypted packets for 20ms network stability and 2ms reconfiguration time for each co-processor unit. The network threshold depends also on the previous and the next configuration. For example, the threshold to change back from the encryption optimized configuration to the balanced configuration is higher (percentage of simple packets must be at least: 49%). Hence, using the specific framework we can easily specify the threshold for each pair of configurations.

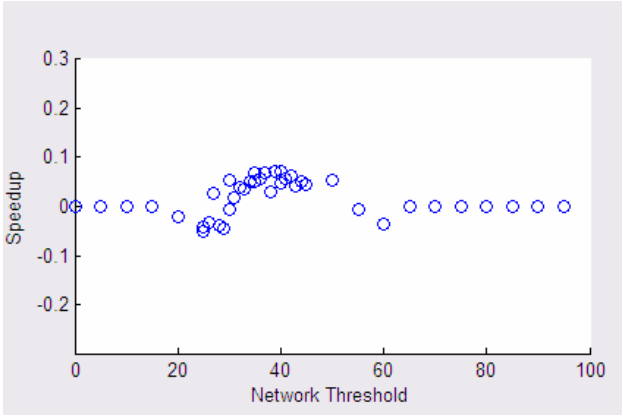


**Figure 6. Design space exploration of threshold**

Besides the network threshold, the sample rate is also crucial to the performance of the system. The sample rate represents the number of packets that have to be received to measure the network distribution. If the sample rate is too high (e.g. every 50 packets) then there is no accurate estimation of the network distribution. On the other hand, if the sample rate is too low (e.g. every 400 packets) then the system fail to adapt fast to the changing network traffic hence, the speedup is reduced. Figure 7 shows the design space exploration of the sample rate (number of packet to be accumulated) for 20ms network stability and 2ms reconfiguration time. The figure shows that the maximum speedup is achieved using around 100 packets. The overhead of the sample rate is usually negligible to the performance of the system. After the classification of the packet a counter that counts the number of packets that belong to each flow is upgraded without wasting processing power.

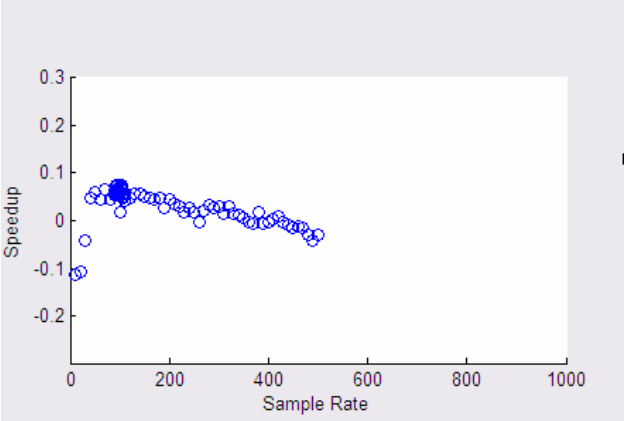


**Figure 7. Design space exploration of sample rate**

Finally the framework is used to perform a design space exploration for various network threshold and sample rates. Figure 8 shows the speedup of the system for several sample rates (30, 60, 90, 120 and 150 packets) and for several network thresholds (30%, 35%, 40%, and 45%). As it is shown both of these parameters are crucial to the speedup of the systems. The figure shows the design space exploration for 20ms network stability and 2ms reconfiguration for each co-processor unit. The best speedup is performed when the sample rate is around 120 network packets and the minimum threshold before the reconfiguration is 40% of a specific flow. Hence, using this framework we can easily tune the configuration manager depending on the network characteristics in order to achieve the maximum speedup of the reconfigurable platform.

The main advantage of the proposed design flow is that a simple Look-Up-Table can be constructed in an embedded RAM (Block RAM) that stores the performance of several configurations such as in Table 2 and decide on the reconfiguration based on the network stability and using the equation shown in (1). The network stability can be obtained by counting the time that the distribution of the flows remains the same. Hence, the configurable manager can achieve efficient decision making about the scheduling of the reconfigurable units by using this LUT without performing complicated algorithms.

## V. CONCLUSIONS

In this paper an integrated framework has been presented that can be used to perform a design space exploration of the performance of a reconfigurable platform for network processing. This framework can be used to easily evaluate if the reconfiguration can be used in specific network application and network traffic. Furthermore, it provides the ability to perform a design space exploration in order to tune the reconfiguration manager in terms of threshold and sample rate. These parameters are crucial to achieve the maximum speedup over a static version. Finally, a case study is presented in which a network system has to process three flows with different processing requirements and how the dynamic reconfiguration can be exploited to increase the performance of the system.

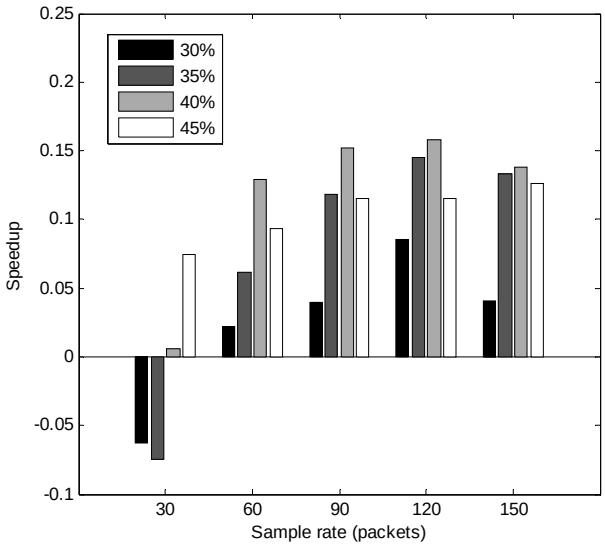


**Figure 8. Design space exploration for sample rate and threshold**

## ACKNOWLEDGMENT

This work was supported by Sandbridge Technologies, Inc.

## REFERENCES

[1] 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

[2] 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

[3] 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

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

[5] 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.

[6] C. Kachris, S. Vassiliadis, "Analysis of a Reconfigurable Network Processor", *Reconfigurable Architectures Workshop, IEEE International Symposium on Distributed and Parallel Systems*, Rhodos, Greece, April 2006

[7] C. Kachris, S. Vassiliadis, "Performance Evaluation of an Adaptive FPGA for Network Processing", *IEEE International Conference on Rapid Systems Prototyping*, Chania, Greece, June 2006

[8] K. Papademetriou, A. Dollas, “Performance Evaluation of a Preloading Model in Dynamically Reconfigurable Processors”, IEEE *International Conference on Field Programmable Logic and Applications*, Madrid, Spain, August 2006

[9] K. Papademetriou, A. Dollas, “A Task Graph Approach for Efficient Exploitation of Reconfiguration in Dynamically Reconfigurable Systems”, *IEEE Symposium on Field Programmable Custom Computing Machines*, San Jose, CA, April 2006

[10] J. Resano, D. Mozos, F. Catthoor, D. Verkest, “A Reconfiguration Manager for Dynamically Reconfigurable Hardware”, *IEEE Design and Test of Computers*, Sept.-Oct. 2005, Vol. 22, Issue: 5, pp. 452- 460

[11] Z. Li, S. Hauck, “Configuration Prefetching Techniques for Partial Reconfigurable Coprocessor with Relocation and Defragmentation”, *International Symposium on FPGAs*, ACM Press, pp. 187-195, April 2002

[12] T. O’Sullivan, R. Studdert, “Configuration Management for Networked Reconfigurable Embedded Devices”, *Mobility Aware Technologies and Applications*, Springer-Verlag, pp.98-107, January 2005

[13] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K.L.M. Bertels, G.K. Kuzmanov, E. Moscu Panainte, “The Molen Polymorphic Processor”, *IEEE Transactions on Computers*, pp. 1363- 1375, November 2004, Volume 53, Issue 11

[14] 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

[15] R. Maestre et al., “A Framework for Reconfigurable Computing: Task Scheduling and Context Management”, *IEEE Transactional on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No.6, December 2001

[16] Xilinx Power Consumption Tools, “Virtex 4 XPower Estimator spreadsheet”, www.xilinx.com

[17] A. Raghunath, A. Kunze, E. J. Johnson, V. Balakrishnan, “Framework for supporting multi-service edge packet processing on network processors”, Symposium on Architectures for Networking and Communications Systems, Princeton, NJ, October, 2005