The Dissertation Committee for Byeong Kil Lee
Certifies that this is the approved version of the following dissertation:

Network Processor Design: Benchmarks and Architectural Alternatives

Committee:

Lizy K. John, Supervisor

Ronald Barr

Chen-Chau Chu

Sanjay Shakkottai

Earl E. Swartzlander, Jr.

Network Processor Design: Benchmarks and Architectural Alternatives

by

Byeong Kil Lee, B.S.E, M.S.E.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2005

To my family and friends

Acknowledgements

I would like to thank my advisor, Dr. Lizy John for her advice, guidance, and support. She has had a significant influence not only as my graduate advisor but also on my life. Her professional integrity and pursuit of perfection helped me become a better individual. Her thoughtfulness and understanding the status of international students made me have more freedom towards research without any financial problem. I am grateful to her for unconditional support and encouragement she gave me throughout my Ph. D. study. I'm also grateful to her guiding me to have great experiences on professional activities.

I also thank my committee, Prof. Ronald Barr, Dr. Chen-Chau Chu, Prof. Sanjay Shakkottai, and Prof. Earl E. Swartzlander Jr. for their invaluable comments, productive suggestions, and the time to reading my thesis.

I would like to thank the students (past and current) at the Laboratory for Computer Architecture (LCA) - Deepu Talla, Ravi Bhargava, Juan Rubio, Madhavi Valluri, Tao Li, Rob Bell, Yue Luo, Shiwen Hu, Anand Rajan, Aashish Phansalkar, Ajay Joshi, Sean Leather, Lloyd Bircher, Hari Angepat, Jason Matalka and Hareesh Pottamsetty. They have contributed to my research by providing valuable comments on drafts of my paper submissions and useful feedback at practice talks.

I would like to thank Dr. Thang Tran, Teik Tan, Ty Garibay and Dr. Jim Bondi for providing me with an opportunity to work at Texas Instruments, Austin. I also thank all team members – Bhasi, Sam, James, Abraham, Paul, Sree, Jeff, Arjun, Murali, Rajinder, Rao, Bob, Hongjoong and Bill.

Thanks to Linda, Shirley, Debi, Melanie, Amy and other administrative assistants who worked in Computer Engineering in the past years.

Finally, I would like to thank my parents, wife, Jisop (son), Hayoung (daughter) and friends who have had a tremendous influence on my life.

BYEONG KIL LEE

The University of Texas at Austin

August 2005

Network Processor Design: Benchmarks and Architectural Alternatives

Publication No._____

Byeong Kil Lee, Ph. D.

The University of Texas at Austin, 2005

Supervisor: Lizy John

The last decade saw phenomenal growth in information technology and network communication. The network interfaces also have to keep up with the speed, throughput and capability to support all the workloads. Network processors (NPs) have recently been introduced in the network interfaces to process complex workloads.

This dissertation investigates architectural alternatives for network processors. The network processor should be able to process modern network workloads without slowing down line speed. In order to handle variety of emerging applications, good understanding of the target application from the architectural perspectives is essential. While most of the previous research and commercial products for NPs are dedicated to routing and communication related to data-plane applications, control-plane applications where congestion control and QoS issues are dealt with are not well understood. With the

demands of emerging network applications, it is imperative to develop and quantitatively characterize the NP control plane workloads to guide architects for designing future NPs.

In this dissertation, a new benchmark suite, called ***NpBench***, is proposed for network processors and its architectural workload characteristics are studied. The NpBench suite includes 5 control plane applications and 5 data plane applications. The NpBench suite is implemented using C and is opened to public. Large number of institutions in the world has licensed and several papers and articles cite the NpBench. The NpBench suite fills a major void that exists in the evaluation and benchmarks of NPs.

Another major contribution of this dissertation is architectural enhancements for network processing. First the parallelism characteristics of network processing applications were investigated to see the possibility of identifying it statically. Based on the investigation, it is found that the success of VLIW in the multimedia field can be applied to the network processor domain as a processing element for a parallel architectural implementation.

As alternative solutions of existing network processor architectures, hardware acceleration techniques are proposed to deal with new emerging workloads. Also, the feasibility of extracting common ISA extensions over variety of network workloads is investigated for accelerating the capability of a processing element within a parallel architecture.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The last decade saw phenomenal growth in information technology and network communication. As Internet and network technologies have exponentially grown and evolved, the requirement of network interface has become more complex and diverse. Also, various applications and protocols require more intelligent processing over the network. To keep up with these trends of emerging network applications, programmable microprocessors called the network processor (NP) are introduced in network interfaces to handle the demands of modern network applications.

Several vendors are releasing various network processors using different architectural concepts to meet the features of network application workloads. However, existing network processors are designed for packet processing. Modern network workloads include significant control operations in addition to packet routing. Along with rapidly changing network environments and requirements, alternative architectures to handle emerging network applications are required. This chapter describes (1) the necessity for considering emerging workloads in network processor design, and (2) the objectives and contributions of this dissertation.

1.1 Network Processor Architectures

Network processors can be used in various node positions over the network, such as core, edge and access routers. Core routers (10 Gbps rate) are placed in the middle of

the network, so they are critical for performance and least responsive to flexibility. Edge routers are placed in between core and access devices, requiring medium data rate (2.5 Gbps) and a certain amount of flexibility. URL load balancers and firewalls are examples of edge router functions. Access routers (1 Gbps) provide network access to various devices. Most of their works are related to aggregating and forwarding numerous traffic streams through the network [18][19][74][94][103][112].

The conventional applications of network interface mainly consist of packet processing and classification algorithms. However, modern role of such an interface includes congestion control, network security, accounting [38], network address/protocol translations, load balancing and media transcoding. The processing capability of these emerging workloads must be at a level equivalent to the speed of the network. As a solution to this problem, many NP vendors use the concept of packet-level parallelism (PLP) to satisfy high-performance demands of networks. In fact, various companies use parallel architectures such as single chip multiple processor or fine-grain multithreaded processors to meet the packet-level parallelism [70].

In order to extract the parallelism, it is important to understand the attributes of target applications. More importantly, choosing an appropriate benchmark suite is also a significant factor to successfully design and evaluate the processors.

While GPPs (General Purpose Processors) are flexible to rapidly developing network applications and protocols, they do not provide enough performance to process data at wire rates. For example, packet throughput of a 10Gbps link is 19.5 million packets per second, assuming average packet size of bytes [6][45]. Modern network applications require thousands of instructions to be executed per packet in order to accomplish all the

control plane activities required for the packet. General purpose processors at Gigahertz frequencies are not enough to accomplish it. More powerful architectures are required to manage the workloads. Dedicated ASPs (Application Specific Processors) are designed to process packets at wire rates but they are inconvenient when required to add or change the features in order to support new environments. The network processor is a programmable processor or an instruction-set processor specialized for a particular application domain. As shown in Figure 1.1, the network processor exists in the middle range between GPPs and dedicated ASPs.

Due to the variety of application spaces being addressed by network processors, there could be a wide range of NP architectures and implementations. For the enterprise service, several companies developed RISC-based NP with ASIC blocks for networking functions such as IXP 1200/2000 series by Intel [52], CXE-16 by Switchcore, CS2000 by Chameleon etc. For the high-end service, Motorola (C-port) [27], Lucent (FPP/RSP) [71], EZChip (NP-1) [37] and Vitesse/ Sitera(IQ2000) have used network-specific ASICs with the features of network classifying, QoS, etc. Some companies like Chrysalis-ITS, Alliance, NetLogic have developed co-processors with the functions such as routing table, classification or cryptography [96].

Recently, Crowley, *et al*. [28] presented that simultaneous multithreading is best suited for some of the network applications. Recent research and commercial products for network processors show the use of multithreading and vector-type array processing. Melvin, *et al*. [74] utilize multiple multithreaded processing engines to get a high degree of thread-level parallelism (TLP) in an NP design that supports 256 simultaneous threads in eight processing engines. In this scheme, each thread has its own independent register

file, while sharing functional resources and memory ports with other threads. ClearSpeed [24] introduces an MTAP (Multi-Threaded Array Processing) processor, which provides a scalable processing solution, based on an array of 10s to 1,000s of small processing elements. Each PE has its own local memory and I/O capability. Although these implementations can meet the required performance, they have large amounts of hardware and programming complexity, cost and power problems.

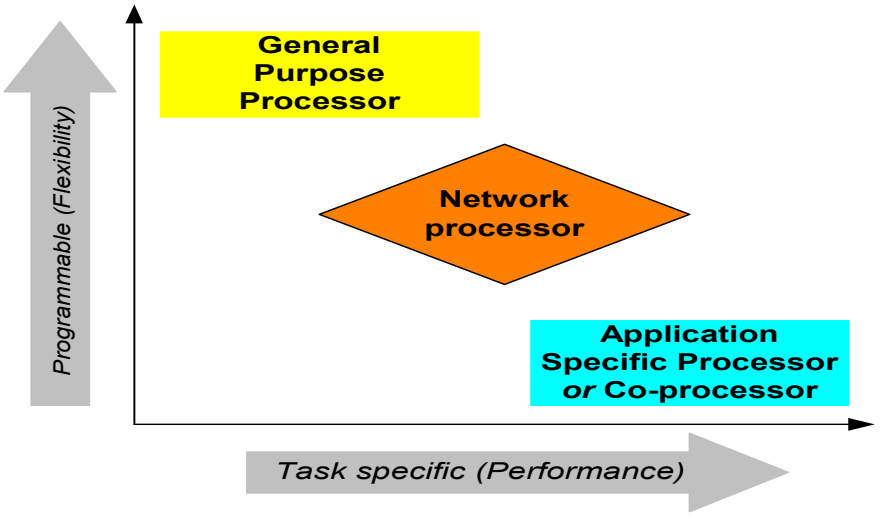


Figure 1.1 Network processors

1.2 NETWORK PROCESSOR WORKLOADS

The bottleneck in communication networks is not just due to bandwidth anymore. Ability to provide flexible processing capability in order to support several emerging ap-

plications and meet their heavy processing workloads is equally important [64]. Major challenges for high bandwidth have reached tremendous advances from optical network approaches, a solution to bandwidth-centric bottleneck – currently 10Gbps (OC-192) at core router exists and 40Gbps (OC-768) is now starting to emerge. In modern network areas, more complex protocols and various network services (e.g., Quality of Service, IPSec, IPv6, etc.) require significant processing power for highly intelligent applications, so the bottleneck of communication network has moved to the network nodes. Accordingly, extracting representative benchmarks from wide range of emerging network applications and characterizing their properties are essential for designing network processors and evaluating their performance.

NP applications can be functionally categorized into two types of operations: data plane operations and control plane operations. The data plane performs packet forwarding, fragmentation and checksum calculation. The control plane handles flow management, signaling, higher-level protocol and other control tasks [3][26][64]. Over the past few years, several vendors have been releasing NPs using a number of different architectures, but most of them are optimized for throughput mostly in data plane. Also, existing benchmark suites for network processors primarily contain data plane workloads, which perform packet processing for a forwarding function. Although NPs have initially been targeted for data plane applications as a conventional workload of NPs, they also play a major role in the control plane. In fact, with the increased demand for complex processing, the boundaries between data plane and control plane have become blurred [1]. Control plane operations have become part of most network processor workloads and the significance of control plane has become greater. The recent trend is that some control plane

activities, such as TCP and SSL applications, are being considered as a commodity. Since there are a lot of control mechanisms in TCP, it cannot be easily converted into an ASIC (Application Specific Integrated Circuit) and it has mostly been left to software solutions. The proliferation of control plane workloads has made it clear that control plane operations should be included in NP benchmarks.

1.3 THE PROBLEM

While most of the previous research and commercial products for NPs are dedicated to data-plane applications, control-plane applications are not well understood. With the demands of these emerging network applications, it is imperative to develop and quantitatively characterize the NP control plane workloads to guide architects for designing future NPs.

The network processor on the physical port of a router should be able to process the modern workloads without slowing down line speed. As shown in Table 1.1, computational load per packet ranges from 880 to 4,800, when assuming a stream of minimum-sized packets of 64 bytes and a single processor of 1 GHz clock frequency. The required parallelism (number of instructions per packet) for executing NP applications is in the range of 2 ~ 356 [64]. Hence a conventional processor is not enough to handle these workloads. Relevant research is required to identify appropriate architectures to efficiently execute these emerging workloads [53][69][72][75][87].

Table 1.1 Required parallelism for control plane workloads

Applications	Computational load per packet (number of instructions)	Required parallelism (with a unity ILP 1 GHz processor)		
		1Gbps (~OC-24)	10Gbps (OC-192)	40Gbps (OC-768)
MPLS	3,051	6	60	254
SSLD	4,727	10	94	356
WFQ	2,005	4	40	167
RED	881	2	17	74

1.4 OBJECTIVES

Recent research and commercial products for parallel implementation of network processors are mostly dedicated to data plane applications with multithreaded and vector-type array processors [24][81]. Although these complex hardware implementations can fulfill the demanded performance using massively parallel architectures, they still have problems related to cost of hardware, power, low utilization ratio of processing elements and hardware complexity. This research focus on the following objectives:

- Create a suite of network processor workloads including control plane, so that NPs are designed considering a realistic network workload
- Characterize and understand emerging NP workloads to obtain directions for architecture research
- Investigate NP architectural alternatives for emerging network workloads
- Create hardware accelerators for emerging network workloads

The first objective is to fill the gap that exists in the field of network benchmarks. The objective is to study network algorithms and create benchmarks which can be used in architecture research. Modern network applications can be classified into three functional groups: traffic-management and quality of service group (TQG), security and media processing group (SMG), and packet processing group (PPG). Based on this functional grouping, a set of benchmarks called ***NpBench***, which includes emerging network workloads especially control plane applications, is presented. Control plane workloads are just emerging and evolving in current network environments, and incorporating them in a benchmark suite helps to design future NPs.

To realize the second objective, characterizations of several network workloads on existing processors are performed. The objective is to understand bottlenecks and identify features of the program that can be exploited while designing network processors. From the characterization results of the control plane application, it can be seen that the control plane applications contain large amounts of instruction level parallelism (ILP). The required computational capability for each network application workloads are also evaluated. It is found that aggressive parallel architectures are required in order to meet the required computational performance of the control plane workloads.

The third and fourth objective consists of identifying architectures for emerging network workloads. Network processors can be used in various node positions with different scales, such as core routers (10 Gbps), edge routers (2.5 Gbps) and access routers (1 Gbps). NPs in different positions will need different architectures. Large scale routers may need a very parallel architecture to meet the required throughput, but it is hard to apply such a complex and expensive architecture to small scale routers. Therefore, a sim-

ple and low priced architecture is required for small scale routers to perform the high throughput workloads

In order to understand the design tradeoffs, a fundamental question, whether parallelism should be identified statically or dynamically, is explored. Dynamically scheduled architectures demand less from compiler, however, if the parallelism is easily amenable for compile-time identification, it is a viable alternative. Current popular media processors - TI's C6x and TriMedia's TM-1300 - rely on the simpler hardware of VLIW processors in order to minimize the cost and power of ILP implementation [4][47][54][55][97][113]. This is because multimedia and DSP applications include many loop operations in the algorithm, and they are well suited for the static scheduling architecture. Network processor workloads have also many iterative algorithm and large parallel characteristics, so statically scheduled architectures are worth considering for NP applications. In this approach, characterizing performance and power dissipation of statically identified ILP implementation are performed, and the characterization results are compared to the dynamic optimization for network processor applications.

The possibilities of hardware acceleration as alternatives for emerging control-plane workloads in network processors are investigated. In this dissertation, the characteristics of control plane applications for network processors are analyzed, and application-specific acceleration techniques are proposed to exploit instruction and data level parallelism.

One of the significant application categories is congestion control applications. For this application category, a parallel comparator which decouples serial processing from the queue processing module is applied. The queue processing elements are config-

ured in parallel for exploiting packet level parallelism. In media transcoding application category, most of data to be processed in the transcoding proxy are related to multimedia applications, so each processing unit has regular and fine-grained processing element. Therefore, array-style processor is considered as a candidate for accelerating media transcoding. As a third application category, LUT-related applications are also important workloads in control-plane. Based on the analysis, lookup table searching is a major bottleneck in the LUT-related applications. In order to improve this bottleneck, modified partitioned lookup table mechanism is applied to accelerate lookup table searching module. Parallel comparators are used for equality checking to find exactly matched session IDs in SSLD or FEC (forward equivalence class) ID in MPLS.

1.5 THESIS STATEMENT

Modern network workloads involve significant amounts of control plane operations and require more processing than the capability of general purpose processors. Hardware acceleration approaches can deliver the required processing capability at low cost.

1.6 CONTRIBUTIONS

This dissertation makes several contributions to the defining of a new network processor benchmark suite, the characterization of network processor workloads, the detection of bottlenecks in network workloads, and towards designing architectural alterna-

tives including instruction set extensions, hardware acceleration and statically identified parallel architectures. The summary of the contributions is listed below:

1. A benchmark suite, called NpBench targeted towards control plane workloads which are an important part of modern network applications is presented. While previously released network processor benchmarks mainly deal with data plane applications, no benchmark suites are available for control plane workloads. With the increasing demand of QoS [13] and rapidly changing modern network environments, the significance of control plane workloads has become higher. The NpBench suite is implemented using C and is opened to public [64]. Large number of institutions in the world have licensed and several papers and articles cite the NpBench [19][50][77][78][86][88][113][116][118][119]. The benchmark suite is described in [58][64][85].
2. The characteristics of network processor workloads such as instruction mix, cache behavior, available parallelism and required processing capability per packet are presented and compared with existing benchmark suites. Several characterization results including architectural characteristics of the application having control plane properties, their implications to designing network processors and the significance of additional parallelism to perform NP applications at wire speed are described. This contribution is described in more detail in [64].
3. Parallelism characteristics of network processing applications are compared to multimedia applications. NP applications (e.g. WFQ) have both parallelizable operations and an amount of serial operations, while most of multimedia appli-

cations have large amount of parallelizable operations. The analysis of these characteristic differences can be a key to improve the throughput of network processors.

4. It is investigated whether the success of VLIW in the multimedia field can be applied to the network processor domain as a processing element for a parallel architecture. This premise is analyzed through the comparison between network processor workloads and multimedia workloads in terms of performance (speedup) and power consumption. It is found that NP applications need more aggressive optimization techniques in static scheduled architecture, while media applications can get large parallelism with simple basicblock optimizations. With the characteristics of large packet-level parallelism, experimental analysis supports static scheduling as an applicable paradigm for network processor applications with lower hardware complexity and lower power dissipation. This contribution is described in more detail in [59].

5. Congestion control applications contain several parallelizable operations. In order to exploit parallelism of congestion control applications, a hardware acceleration technique is introduced using the decoupling the dataflow into the front-end PLP (Packet level parallelism) module and the back-end hardware acceleration module. This decoupling techniques and defining a hardware acceleration module is done by the thoroughly analysis of the applications. When applying 16 ~ 64 acceleration module in parallel, 10x ~ 50x performance improvement is obtained. This contribution is described in more detail in [65].

6. The procedure of media transcoding consists of transcoding decision unit and pipeline-styled regular processing units for data conversion. The decision module handles which conversion module is enabled and how many modules are enabled. Data conversion modules include several different kinds of functionalities. Based on the experiments, data conversion module takes 89% of the total execution cycles. Most of data to be processed in the transcoding proxy are related to multimedia applications, so each processing unit requires regular and fine-grained processing element. Therefore, the systolic-style processor would be a good candidate for accelerating media transcoding. Array-style acceleration technique is proposed for data conversion module of media transcoding applications. If 64 parallel implementation of the acceleration module is applied, it can be obtained approximately 28x speedup in transcoding. This contribution is described in more detail in [60].
7. Both SSLD and MPLS have the lookup table searching and updating module in each procedure. Lookup table searching is a major bottleneck in the LUT-related applications (77% ~ 86% of the total execution cycles in MPLS and SSLD). In this dissertation, hardware acceleration techniques are proposed using a partitioned lookup mechanism for searching LUT used in MPLS and SSLD. If 16-way parallel implementation of the acceleration module is applied, 262x improvement for MPLS and 362x for SSLD [60] can be obtained.
8. The performance improvement of a single processing element is an important factor in designing parallel architecture. New instruction-set extensions are introduced to support fast execution of congestion control applications, based on

the detailed kernel analysis. Frequently used instructions are combined into one instruction. For congestion control applications, two groups of instruction sets are defined: conditional operations and multiplication-add / multiplication-sub [65]. Proposed architectural extensions show 10~12% improvement in performance for instruction set enhancements.

1.7 ORGANIZATION

The rest of the dissertation is organized as follows:

Chapter 2 describes existing research work pertinent to this dissertation. Previously proposed benchmark suites for network processors are discussed first. Then conventional network processor architectures and industry products are discussed.

Chapter 3 introduces the proposed benchmark suite, called NpBench which includes control plane workloads and data plane workloads. Also, the architectural characteristics of the benchmark are presented and compared to previously released benchmark suite.

Chapter 4 identifies bottlenecks in the execution of network processor applications. This observation is found that the common bottlenecks across both applications are issue width and memory elements.

Chapter 5 discuss whether the success of statically identified parallelism in the multimedia field can be applied to the network processor domain, even though network processor applications have not-so-regular parallel characteristics compared to multime-

dia applications. Effort is also spent in analyzing how these different characteristics in parallelism affect the performance.

Chapter 6 introduces a hardware acceleration technique. Congestion control, media transcoding and LUT related applications are enhanced. For congestion control applications, decoupling the dataflow techniques are introduced. Also, hardware acceleration technique for media transcoding applications and LUT-related applications are presented.

Chapter 7 introduces new ISA extensions to support fast execution of network processing based on the detailed kernel analysis.

Chapter 8 concludes the dissertation by summarizing the contributions and suggesting future opportunities.

Chapter 2: Related Work

Network processors and related workloads have been researched extensively in the past few years. This chapter discusses prior work to this dissertation. The related work includes several different categories: defining benchmarks, characterizing application's workload, and architectural enhancements to get higher throughput.

2.1 Network Processor Benchmarks

In the network processor fields, there are two benchmarks which were previously proposed: CommBench [122] and NetBench [76]. Wolf, *et al*. [122] presented eight selected workloads called CommBench for traditional routers and active routers. CommBench has two groups of benchmarks namely Header Processing Applications (HPA) and Payload Processing Applications (PPA). Memik, *et al*. [76] proposed nine applications called NetBench for micro-level, IP-level and application-level benchmarks. Nemirovsky [80] also discusses the guidelines for defining benchmarks and challenges of benchmark suites for network processors. He suggests that the benchmark should have two frameworks such as a task-specific benchmark focusing on a single algorithm or protocol and a rich-scenario benchmark containing the complexity of real-life applications. EEMBC [36] and MiBench [44] have some network applications, but they only have routing and encryption applications. NpForum [19][82] has released IA (Implementation

Agreement) on IP Forwarding and IPSec Forwarding application level benchmarks, but as obvious, they focus on forwarding.

Previously proposed benchmarks are mainly focused on data plane workloads. While the benchmarks of data plane applications have been reasonably well understood, there has been very little effort in designing control plane workloads that perform congestion control, flow management, higher-level protocols and other control tasks. Control plane workloads are just emerging and evolving in current network environments. The NpBench suite presented in this dissertation is mostly focusing on control plane applications. Also, characteristics of control plane workloads from architectural aspects are analyzed.

2.2 ARCHITECTURAL CHARACTERISTICS OF NETWORK PROCESSOR APPLICATIONS

Past studies using CommBench [122] and NetBench [72] contrast network workloads with other benchmarks, such as SPEC [102] and mediabench [66], with respect to instruction set characteristics and memory behaviors. Wolf, *et al*. [122] indicate that network processors must deal with both streaming and header processing applications. Memik, *et al*. [76] insist that network processor applications have a data-intensive nature. In terms of load/store instruction ratio, the NetBench applications make high number of memory accesses, while MediaBench applications has more frequent branch instructions resulting in a lower instruction level parallelism.

2.3 NETWORK PROCESSOR ARCHITECTURES

In order to deal with variety of application areas for network processors, there could be a wide range of NP architectures and implementations. Each company is applying their own architectural concepts in the implementation of network processors. For the enterprise service, several companies developed RISC-based NP with ASIC blocks for networking functions such as IXP 1200/2000 series by Intel [52], CXE-16 by Switchcore, CS2000 by Chameleon etc. For the high-end service, Motorola (C-port) [27], Lucent (FPP/RSP) [71], EZChip (NP-1) [37] and Vitesse/ Sitera(IQ2000) have used network-specific ASICs with the features of network classifying, QoS, etc. Some companies like Chrysalis-ITS, Alliance, NetLogic have developed co-processors with the functions such as routing table, classification or cryptography [96].

Most of NP architectures employ multiple processing engines (PEs), even though they each have different names such as micro engine, channel processor or task optimized processor. Some are based on RISC cores having their PEs arranged in parallel or in a pipelined fashion. The alternative to the RISC is the VLIW based architecture, in which most of the PEs are organized in a pipelined method. Many RISC based NPs employ multithreading on their PEs to maximize the performance. To ensure fast context switching between tasks, the NP should have hardware support for multithreading. Figure 2.1 shows an overall architecture of typical network processor. In general, control and management functions have more complex processing requirements than data functions. GPPs have been used as control processors in commercial network products. Many NPs provide the function of control processor with an integrated core or externally via a host

interface [49]. In this dissertation, it is shown that GPPs do not have enough processing capability to come up with increased demand for complex processing and higher data rates.

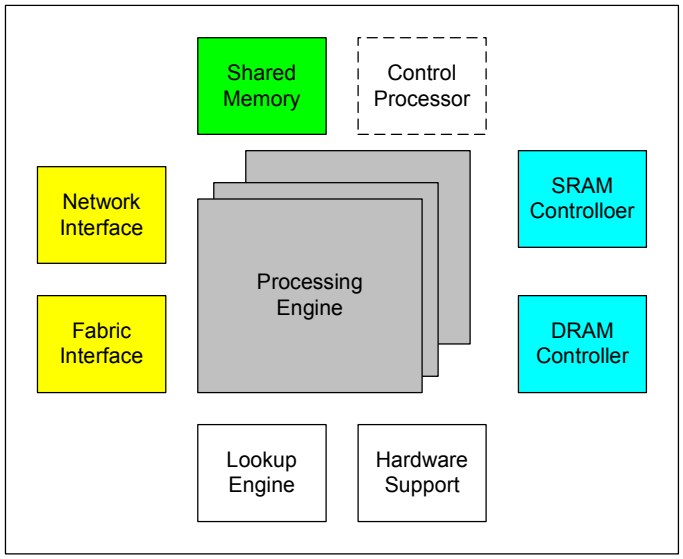


Figure 2.1 Overall architecture of NP

Related recent research and commercial products for network processors show the use of multithreading and vector-type array processing. Melvin, *et al.* [74] utilize multiple multithreaded processing engines to get a high degree of thread-level parallelism (TLP) in an NP design that supports 256 simultaneous threads in eight processing engines. In this scheme, each thread has its own independent register file, while sharing functional resources and memory ports with other threads. ClearSpeed [24] introduces an MTAP (Multi-Threaded Array Processing) processor, which provides a scalable processing solution, based on an array of 10s to 1,000s of small processing elements. Each PE has its own local memory and I/O capability. Although these implementations can meet

the required performance, they have large amounts of hardware complexity, cost and power issues.

Shah, *et al*. [96] indicate that today's network implementations are based on Field Programmable gate Arrays (FPGAs) for lower level processing and General Purpose Processors (GPPs) for higher layer processing. They also investigate the broad categories of alternatives for system implementations such as ASIC (Application Specific Integrated Circuit), ASIP (Application Specific Instruction Processor), Co-processor, FPGA and GPP. They also present the diversity among different network processors. For example, IBM and Motorola have co-processors for most packet-processing kernels, while Cognigine relies on the reconfigurable functional units. EZchip has entire processors devoted to pattern matching, lookup, data manipulation, and queue management. Agere's PayloadPlus system uses a special processor for pattern matching and data manipulation, a co-processor for checksum/CRC computation, and has memory features for queue management. Vitesse and Xelerated Packet Devices simply use a mix of co-processors and functional units. Intel and Lexra also include special memory and bus features and have a dedicated processor for the control-plane.

Chapter 3: Development and Characterization of a Network Processor Benchmark Suite

It is important to have a benchmark suite with emerging network workloads in order to design future network processors. A good benchmark suite must contain emerging and futuristic workloads in order to prevent it from being outdated very quickly. Designing future processors with benchmarks of today, which are programs of yesterday, often results in processors and systems which cannot handle workloads that are prevalent when the processor design is finished.

This chapter presents NpBench, a benchmark suite that was developed to assist designers of future network processor designers. Predictions on future workloads are very difficult to make, however one can examine trends to make educated guesses on what might be probable network workloads a few years from now. The NpBench benchmark suite was created by studying applications from modern network domain workload. A concern that many network processor designers have is that the control plane processing content of network workloads is going up, while the current network benchmarks do not represent that. Hence a study of emerging network applications was conducted, with an emphasis on control plane processing.

3.1 Investigation of Modern Network Workloads

Network applications perform routing, scheduling, traffic management/congestion control, quality of service enforcement, security management, packet processing, etc. A survey of the functionality of modern network applications reveal applications in three major functional groups: traffic-management and quality of service group (TQG), security and media processing group (SMG), and packet processing group (PPG). A categorization of major network applications into these categories is presented in Table 3.1. The table also illustrates whether the application includes control plane content. Since control plane processing is increasing in modern networks, it will be important to include sufficient amount of control plane applications into the benchmark suite.

The investigation of the network workloads unveiled various kinds of applications that in use today or those which are expected to be popular in future. However, not much information is available regarding a mix of these applications in real-world routers. Predicting the mix of applications that will be popular in a network workload a few years from now, is difficult and is outside the scope of this dissertation. The objective of this research will only be to develop source code for a variety of applications that are expected to be popular, and to make it available to other researchers and designers.

It is often impractical to include every application in the world into a benchmark suite. It is often sufficient to choose representative applications for the major classes of applications. In order to decide what applications must be selected into the benchmark suite, information on the workloads used by network routers in the real world was collected. Cisco Systems Incorporation is the biggest router provider and hence an investigation of their applications can provide valuable information on what is important in current

and future network environments. Information from CISCO [23][21], Lightsurf [73] and IBM [8] was used in making some deductions on emerging network workloads.

Table 3.1 Functional grouping of network processor workloads

Group	Applications	Data Plane	Control Plane
TQG (Traffic-management and Quality-of-Service Group)	Routing	X	X
	Scheduling	X	X
	Content-based switching	X	X
	Weighted fair queuing	X	X
	Traffic shaping	X	X
	Load balancing	X	X
	VLAN		X
	MPLS	X	X
	RSVP	X	X
	DiffServ	X	X
	IntServ	X	X
SMG (Security and Media Processing Group)	Block cipher algorithm	X	
	Message digest algorithm	X	
	Firewall application	X	X
	IPSec	X	X
	Virtual private network	X	X
	Public encryption	X	
	Usage-based accounting	X	X
	H.323	X	
	Media transcoding	X	X
	Duplicate data suppression	X	
PPG (Packet Processing Group)	IP-packet fragmentation	X	
	Packet encapsulation	X	
	Packet marking/editing	X	
	Packet classification	X	
	Checksum calculation	X	

Weighted fair Queuing (WFQ) is one of Cisco's premier queuing techniques [23]. For situations in which it is desirable to provide consistent response time to heavy and

light network users without adding excessive bandwidth, the solution is WFQ. It is a flow-based queuing algorithm, and it can work in conjunction with RSVP (Resource Reservation Protocol) to build Integrated Service architecture implementations which are designed to guarantee network bandwidth from end to end for IP networks. Hence WFQ was included in the NpBench suite.

Random Early Detection (RED) is useful on any output interface where expected to have congestion. RED is usually used in the core routers of a network, rather than the network's edge. Edge routers assign IP precedences to packets as they enter the network. Cisco's RED implementations include Distributed Weighted Random Early Detection [23], which combines the capabilities of the RED algorithm with IP Precedence. This combination provides for preferential traffic handling for higher priority packets. WRED provides separate thresholds and weights for different IP precedences, which can provide different qualities of service for different traffic. It is not known whether RED will be used heavily in the future, but RED is included in the suite because if traffic management becomes critical, RED is likely to be employed.

Cisco's IOS Multiprotocol Label Switching (MPLS) enables Enterprises and Service Providers to build next-generation intelligent networks that deliver a wide variety of advanced, value-added services over a single infrastructure [21]. This economical solution can be integrated seamlessly over any existing infrastructure, such as IP, Frame Relay, ATM, or Ethernet. Subscribers with differing access links can be aggregated on an MPLS edge without changing their current environments, as MPLS is independent of access technologies. Integration of MPLS application components, including VPN, Traffic Engineering, QoS and IPv6 enable the development of highly efficient, scalable, and se-

cure networks that guarantee Service Level Agreements. MPLS appears to be a likely component of future network workloads and hence it is included in the suite.

Another popular application appeared to be SSL (Secure Socket Layer), the de-facto standard in securing distributed applications [8]. Originally defined by Netscape Communications, SSL is accepted for network applications for authenticated and encrypted communication between clients and servers. The SSL protocol runs above TCP/IP and below higher-level protocols such as HTTP. It uses TCP/IP on behalf of the higher-level protocols, which allows an SSL-enabled server to authenticate itself to an SSL-enabled client, allows the client to authenticate itself to the server, and allows both machines to establish an encrypted connection. These capabilities address fundamental concerns about communication over the Internet and other TCP/IP networks. SSLD is a technology to reduce the load of authentication between server and client.

A significant part of network workloads appear to be handling multimedia data. Often, data exists in one format at a server, and has to be displayed in a simpler format on a mobile device. Media transcoding (MTC) delivers optimized pictures and other multimedia content across wide range of heterogeneous devices. LightSurf [73] provides a media transcoing product which senses what kind of viewing device the recipient is using and intelligently transcodes the images and other multimedia files for optimal delivery. Throughout the world, there are hundreds of different graphics-capable mobile phones that all have varying display characteristics. Cisco [23] also provides Media Resource Manager (MRM) as resource reservation of transcoders within a Cisco CallManager cluster. Cisco CallManager supports simultaneous registration of both Media Transfer Part

(MTP) and transcoder and concurrent MTP and transcoder functionality within a single call.

Security is another major issue in networks and security applications are expected to be a major component of future network workloads. IPSec is a framework of open standards developed by the Internet Engineering Task Force (IETF) that provides security for transmission of sensitive information over unprotected networks such as the Internet [23]. In the case of secure VPN client IPSec, as the tunnel comes up, the PC receives its IP address from the central router's IP address pool, then the pool traffic can reach the local or be routed and encrypted to the network behind the outlying router. IPSec provides a more robust security solution and is standards-based. IPSec also provides data authentication and anti-replay services in addition to data confidentiality services.

Packet processing applications are essential and an important part of modern network workloads. Every packet-based network has an MTU (Maximum Transmission Unit) size. The MTU is the size of the largest packet network can transmit. Packets larger than the allowable MTU must be divided into multiple smaller packets, or fragments, to enable them to traverse the network. Packet fragmentation appears to be an essential application for packet processing.

Error detection and correction is an important part of network processing. Cycle redundancy check (CRC) is a technique used to check errors. The CRC uses a calculated numeric value to detect errors in transmitted data. The sender of a data frame calculates the Frame Check Sequence (FCS). The sender appends the FCS value to outgoing messages. The receiver recalculates the FCS, and compares it with the FCS from the sender. If a difference exists, the receiver assumes that a transmission error occurred, and

sends a request to the sender to re-send the frame. The retention of the true value of a frame is important to ensure that the destination correctly interprets the data over the network. CRC is another essential application for packet processing.

Based on this analysis, ten representative applications are chosen from the functional groups for the first version of NpBench suite as shown in Table 3.2. The suite includes several control plane functions as they are missing from the available NP workloads.

Table 3.2 Benchmarks in the NpBench suite

Group	Application	Description
TQG	WFQ	*Weighted Fair Queuing* is a queue scheduling algorithm
	RED	*Random Early Detection* is an active queue management algorithm which drops arriving packets probabilistically
	SSLD	*Secure Sockets Layer Dispatcher* is an example of content-based switching mechanism
	MPLS	*Multi Protocol Layer Switching* is a forwarding technology using short labels
SMG	MTC	*Media Transcoding* is the process that a media object in one representation is converted into another representation for wide spectrum of client types
	AES	*Advanced Encryption Standard* (RijnDael) is a block cipher that encrypts and decrypts 128, 192 and 256 bits blocks
	MD5	*Message Digestion* algorithm takes as input a message of arbitrary length and produces as output a 128-bit fingerprint or message digest of the input
	DH	*Diffie-Hellman* key exchange allows two parties who have not met to exchange keys securely on an unsecure communication path
PPG	FRAG	*FRAG* is a packet fragmentation application
	CRC	*Cyclic Redundancy Check* is used in Ethernet and ATM Adaptation Layer 5 (AAL-5) checksum calculation

Some of these selected applications are implemented to form the current release of the NpBench suite and the rest of them are referred from open source code sites or other benchmarks [44][66][122]. The C code for the benchmarks is available on request [85].

3.2 DESCRIPTION OF THE APPLICATIONS IN THE NPBENCH SUITE

3.2.1 Traffic-management and QoS Group (TQG) Benchmarks

TQG benchmarks have a set of applications related to routing, scheduling, queuing, switching, signaling and quality of services. These applications contain both control plane processing and data plane processing. The first two benchmarks, WFQ and RED are congestion control algorithms. In general, congestion occurs at a router when incoming packets arrive at a rate faster than the rate the router can switch them to an outgoing link. The two representative algorithms for congestion control are the scheduling algorithm and the queue management algorithm [15][39]. The scheduling algorithm determines which packet to be sent next and is used primarily to manage the allocation of bandwidth among flows (e.g., weighted fair queuing). According to the IETF (Internet Engineering Task Force) recommendation [15], the default mechanism for managing queue lengths to meet these goals in FIFO queues is the RED algorithm. SSLD is a content-based switching algorithm and MPLS is a technology used for quick forwarding of packets across backbones.

WFQ (Weighted Fair Queuing): WFQ [11][14][30][105] is a queue-scheduling algorithm to serve packets in order of their finish-times considering the weight on connections. As shown in Figure 3.1, various lengths of packets from incoming traffic are classified into different queues, which can be used for differential service. And they are scheduled by a specific mechanism that determines packets to be sent from the queues. WFQ uses each packet's estimated finish-time to decide packets to be sent.

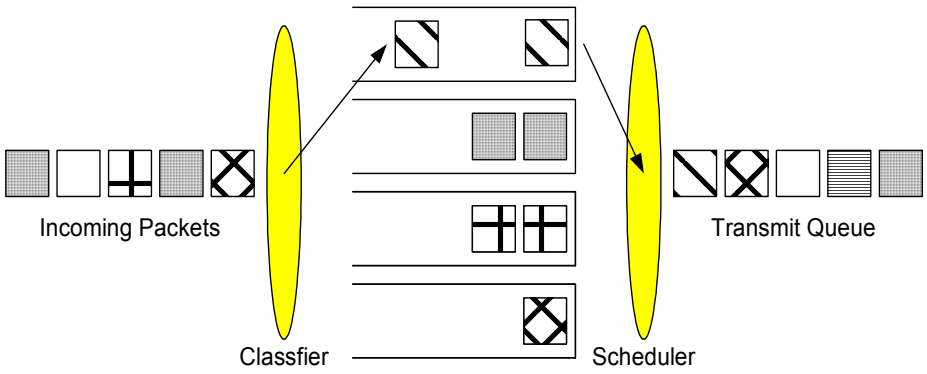


Figure 3.1 WFQ (Weighted Fair Queuing)

RED (Random Early Detection): RED [15][39] is an active queue management algorithm for routers. In contrast to the traditional queue management algorithm, which drops packets only when the buffer is full, the RED algorithm drops arriving packets

probabilistically before coming into the queue as shown in Figure 3.2. The decision of whether or not to drop an incoming packet is based on the estimation of the average queue size.

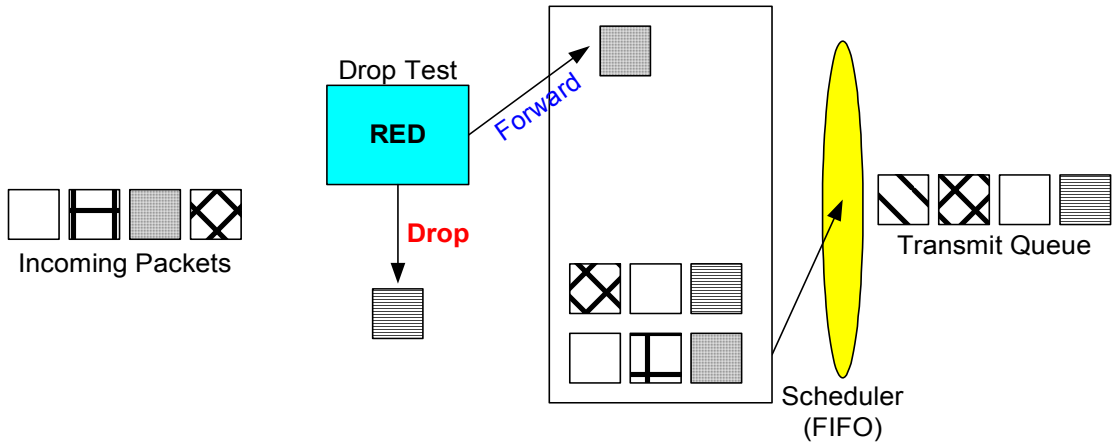


Figure 3.2 RED (Random Early Detection)

SSLD (SSL Dispatcher): SSLD [7][99][102] is one example of content-based switching mechanism in the server and client cluster environments. SSL typically runs over TCP (Transmission Control Protocol), which is used for secure processing of e-commerce applications. Once TCP connection is established, SSLD maintains the session ID information during authentication process, sharing the SSL information among the nodes in cluster. When reconnecting to the same server, a client can reuse the session

state established during a previous SSL handshake which makes the workloads computationally heavy. Figure 3.3 shows the concept of SSL dispatcher.

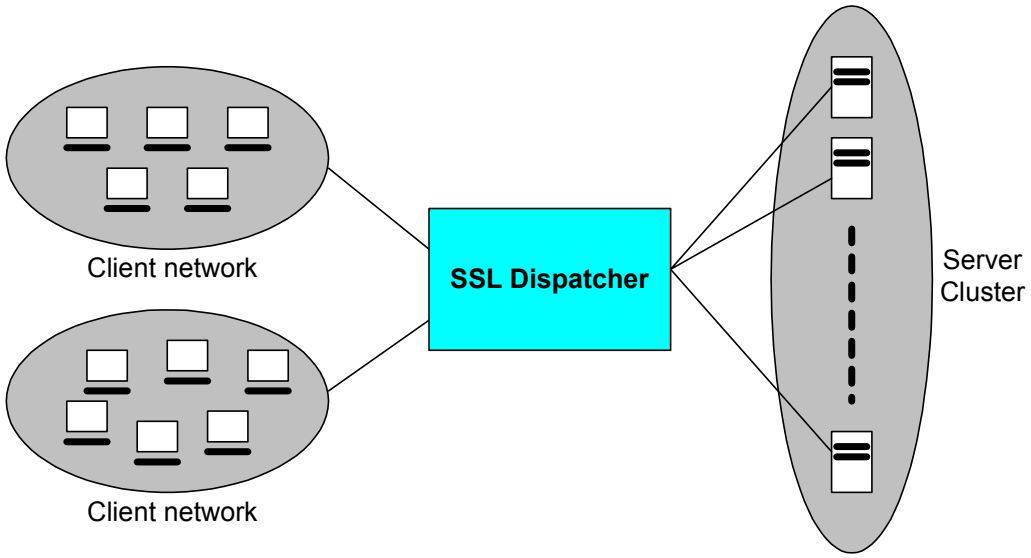


Figure 3.3 SSLD (SSL Dispatcher)

MPLS (Multi Protocol Label Switching): MPLS [9][56][79][93][126] is a forwarding technology, which does away with the lookup of bulky IP headers and uses short labels for forwarding at the edge of the MPLS domain as shown in Figure 3.4. In this version of NpBench, two control plane aspects of MPLS such as Label Distribution and Label Generation, are used. Two functions are extracted from MPLS, namely an upstream routing function (for an ingress edge router or a core router) and a downstream routing function (for a core router or an egress router).

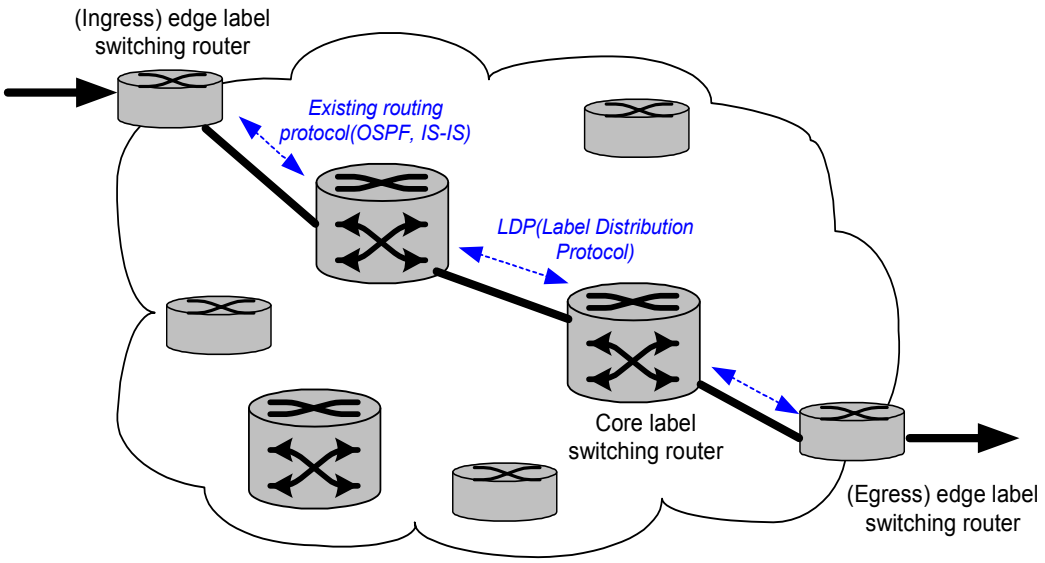


Figure 3.4 MPLS (Multi Protocol Label Switching)

3.2.2 Security and Media processing Group (SMG) Benchmarks

As the e-commerce industry has grown, the security and the accounting applications such as firewall application, admission control, encryption applications and usage based accounting, have become an emerging workload. With higher bandwidth, the demand for high quality of multimedia service is increased. Data stream manipulation, media transcoding, H.323 and several encoding applications [61][62] can be important issues of NP, associated with QoS. For security benchmarks, three components of IPSec

[19][33][95] – Authentication Header (AH), Encapsulating Security Payload (ESP) and key management – are included in SMG.

MTC (Media TransCoding): Media Transcoding [46][99] is a process in which a data object in one representation is converted into another representation. In order to accommodate the wide spectrum of client capabilities, the media data is modified along the dimensions, fidelity, and resolution. As shown in Figure 3.5, media transcoding consists of transcoding policy decision module and transcoding module. The trnascoding policy decision module decides which factors should be converted to another representation based on client device information, contents information and network environments. The transcoding module deals with actual conversion processing.

AES (Advanced Encryption Standard): Advanced Encryption Standard (RijnDael) [1][44] is a block cipher that encrypts and decrypts 128, 192 and 256 bit blocks, which is a U.S. government standard for encryption and digital signature. It is used for implementation of AH in IPSec. AES, designed by Joan Daemen & Vincent Rijmen, is a block cipher using symmetric key. It is fast and scalable, and it is also resistant to all known cryptanalysis attacks. Decryption is 30% slower than encryption, since inverse matrix calculation is more complicated. In the algorithm, the block is considered to be structured as 4, 6 or 8 columns of 4 bytes, depending on block size. The basic operations applied to the block are KeyAddition, Substitution, ShiftRow and MixColumn.

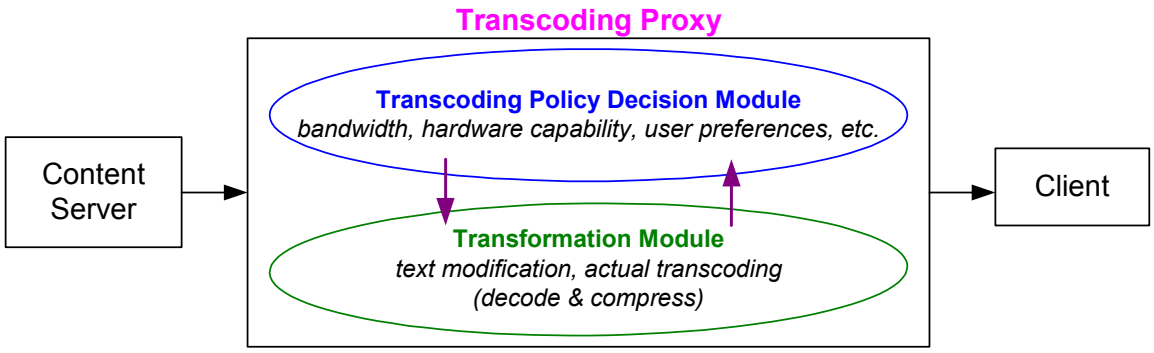


Figure 3.5 MTC (Media Transcoding)

MD5 (Message Digestion): MD5 algorithm [44][122] takes a message of arbitrary length as an input and produces a 128-bit “fingerprint” or “message digest” as an output. MD5 is a method to verify data integrity and is more reliable than checksum method. It is used to perform ESP in IPSec. An algorithm created in 1991 by Professor Ronald Rivest that is used to create digital signatures. It is intended for use with 32 bit machines and is safer than the MD4 algorithm, which has been broken. MD5 is a one-way hash function, meaning that it takes a message and converts it into a fixed string of digits, also called a message digest. When using a one-way hash function, one can compare a calculated message digest against the message digest that is decrypted with a public key to verify that the message hasn't been tampered with. This comparison is called a "hashcheck."

DH (Diffie-Hellman): Diffie-Hellman [44][76][122] key exchange allows two parties who have not met, to exchange keys securely on an unsecured communication path. Typically DH is used to exchange a randomly generated conventional encryption key, the rest of the exchange is then encrypted with the conventional cipher. It is applied to the function of key management in IPSec. Diffie-Hellman key exchange allows two parties who have not met to exchange keys securely on an unsecure communication path. It has been used with DES, 3DES, IDEA, RC4 though basically the approach of using DH key exchange can be used for any conventional stream or block cipher. PGP itself operates in a similar fashion, except that PGP uses RSA for key exchange, and IDEA as the conventional cipher.

3.2.3 Packet Processing Group (PPG) Benchmarks

Packet processing group includes IP packet fragmentation, packet marking, editing and classification. Most applications are data plane processing.

FRAG (Packet Fragmentation): FRAG [122] is a packet fragmentation application. IP packets are split into multiple fragments for which some header fields have to be adjusted and a header checksum computed.

CRC (Cyclic Redundancy Check): 32-bit Cyclic Redundancy Check [44] is used in Ethernet and ATM Adaptation Layer 5 (AAL-5) checksum calculation.

In summary, the NpBench suite consists of both essential workloads widely used in real world and emerging workloads starting to be used in real applications or services. Some of control plane workloads including WFQ, RED and MPLS are used in real world, while SSLD and MTC are emerging workloads which will be significant workloads in routers. These emerging applications will be widely used at routers as Internet business and wireless technologies are evolving. In the data plane workloads, three IPSec applications are very important in all security-related areas including banking, personal identification and e-commerce applications. FRAG and CRC are basic and essential operations for packet processing.

3.3 IMPLEMENTATION

The NpBench control-plane functions such as WFQ, RED, SSLD, MPLS and MTC at the application level are developed using C language, and most of the data-plane functions such as FRAG, CRC and IPSec applications were extracted from pen source code or other benchmarks [44][66][76][122].

Randomly generated packets are used as input of the benchmark. For TQG benchmarks, WFQ uses packet size and queue information as an input. RED uses incoming packet size and average queue size to decide whether the packet is to be dropped or put in the FIFO queue. The clientHello message and serverHello message of the SSL protocol [102] are used with randomly generated session ID information for the experiment of SSLD. The values of FEC identification number are used for two MPLS functions. The RED implementation allows an option of congestion environment, which is con-

trolled by transmission rate of the queue. The SSLD inputs can be different session IDs with different reusability factors. In SMG benchmarks, MTC can be separated into two components which are policy modules to get adaptive transcoding policies and transformation modules to perform real transcoding. The policy decision module can be executed independently with an execution option. AES can use any files as an input data and MD5 can make a fingerprint of any files or strings for the input. DH generates and exchanges for any number of Diffie-Hellman key pairs. For PPG benchmarks, FRAG and CRC employ randomly generated IP header as an input data. Under the above simulated network environments, the characteristic of NpBench is investigated and presented at next section.

3.4 ARCHITECTURAL CHARACTERISTICS OF NPBENCH WORKLOADS

In this section, experimental results including instruction distribution, cache behavior and parallelism of the NpBench are presented. These metrics are essential information for understanding dynamic characteristics of the application and for designing the processor architecture. Also, required computational capability (in terms of number of instructions) to process one packet data is explored, assuming a minimum-sized packet of 64 bytes.

3.4.1 Experimental Methodology

The SUN Shade binary instrumentation tool [25] is used to obtain the dynamic traces while executing NpBench applications. Cachesim5, a cache analyzer tool of SUN

Shade [25], is also used to perform cache simulation and Tetra [10] to get available parallelism with constraints.

3.4.2 Instruction Distribution

In these experiments, the number of instructions in the NpBench applications is investigated. Table 3.3 shows the dynamic instruction distribution during execution. From this workload distribution, it is observed that computational operations occupy a significant share of the total instruction mix (53% on the average). Branch operations (branch, jump and call) are heavily used in the applications having control plane functions (23.7%) such as WFQ, SSLD and MPLS, for finding fair conditions of each packet, looking up session reuse conditions of each session request and investigating same forwarding equivalence class respectively. Data plane functions have relatively small percentage of branch operations (11.1%).

Since the data plane application is to handle more packet data and coefficients for performing the algorithm within payload processing, it is found that the data plane application (31.2% on average) uses more memory operations (load and store) than the applications having control plane functions (23.5%). In the case of SSLD, as the reusability factor used in SSLD increases, it is observed that the required computation workloads for new session request could be avoided and the required number of instructions could be reduced.

Compared to CommBench, the NpBench has similar percentage of ALU operations out of the total instructions. However, branch operations are heavily used in

NpBench control plane applications (23.7%), followed by CommBench-HPA (20.6%), CommBench-PPA (15.3%) and NpBench data plane applications (11.1%).

Table 3.3 Instruction distribution

(Unit: %)

NpBench							
App.	**int/float**	**shift**	**logic**	**branch**	**load**	**store**	**etc**
WFQ	20.6	16.9	0.0	29.2	16.2	7.9	9.3
RED	39.7	7.2	0.0	15.3	23.6	10.9	3.0
SSLD	57.0	0.0	0.0	28.3	14.4	0.2	0.0
MPLS	35.8	11.4	8.6	22.0	16.1	4.9	1.3
MTC	33.9	2.6	10.5	11.2	21.7	18.1	1.9
AES	10.5	18.4	26.9	7.0	29.4	7.7	0.1
MD5	45.1	13.0	20.5	7.5	7.1	6.7	0.2
DH	24.0	12.0	10.9	9.7	27.0	11.3	5.1
CRC	25.0	10.0	15.0	10.0	25.0	15.0	0.0
FRAG	40.0	3.8	15.1	21.4	12.2	6.1	0.7
Avg.	*33.2*	*9.5*	*10.8*	*16.2*	*19.3*	*8.9*	*2.2*

(a) NpBench

(Unit: %)

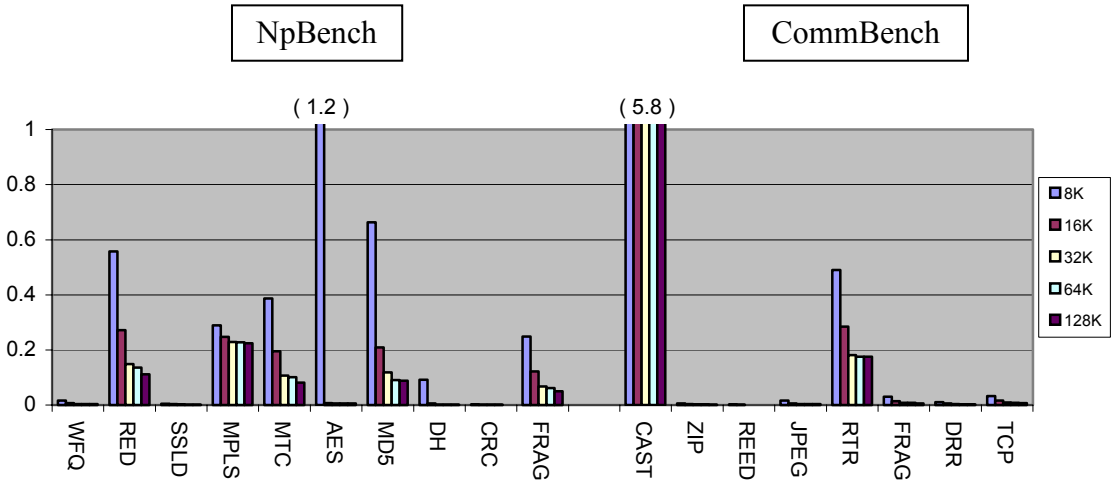
CommBench							
App.	**int/float**	**shift**	**logic**	**branch**	**load**	**store**	**etc**
CAST	25.4	17.0	20.4	8.9	20.4	7.4	0.5
ZIP	34.0	8.0	12.4	20.2	19.4	5.6	0.4
REED	40.2	11.7	7.1	21.4	14.7	4.9	0.0
JPEG	43.8	16.1	2.7	10.8	16.5	9.7	0.3
PPA	*35.8*	*13.2*	*10.7*	*15.3*	*17.8*	*6.9*	*0.3*
RTR	20.6	0.7	11.0	23.4	41.3	2.7	0.2
FRAG	41.5	3.8	15.0	20.4	12.8	6.5	0.0
DRR	31.7	1.0	0.2	18.3	41.8	6.9	0.1
TCP	37.2	5.2	12.5	20.5	16.4	7.1	1.3
HPA	*32.8*	*2.6*	*9.7*	*20.6*	*28.1*	*5.8*	*0.4*
Avg.	*34.3*	*7.9*	*10.2*	*18.0*	*22.9*	*6.3*	*0.4*

(b) CommBench

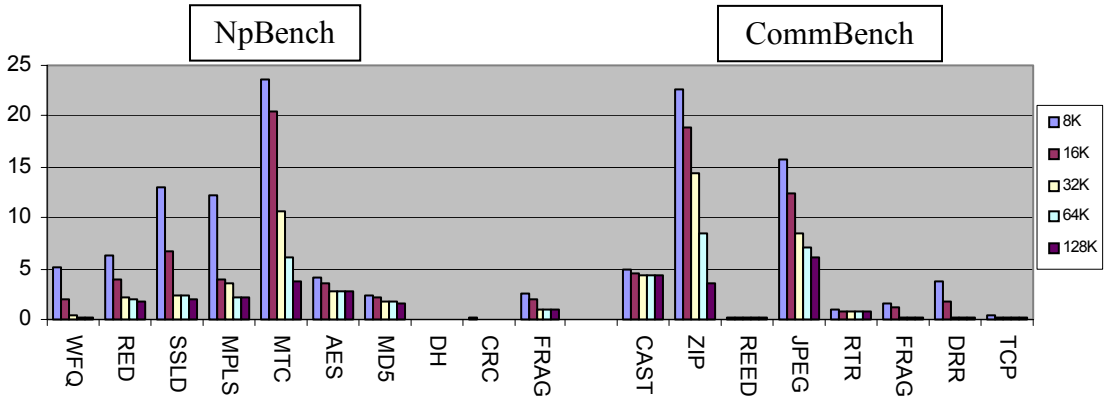
3.4.3 Cache Behavior

The general purpose processor spends a significant part of their real estate for on-chip caches and hence it is important to understand cache behavior of the NpBench applications executing on GPP. Cache performance for 2-way set associative cache was evaluated with varying cache sizes. A line size of 32 bytes was commonly used for all cache configurations. Figure 3.6 shows the cache miss rates for NpBench applications. Most of the NpBench applications perform same operations with various inputs, explaining the excellent instruction cache hit ratios. However, data cache performance of these applications is very poor for small cache sizes. The average miss ratios converge to 0.056% for I-cache and 1.531% for D-cache with increasing cache sizes. Instruction cache sizes larger than 16KB marginally increase cache performance and same observations are made with data cache sizes larger than 32KB. This implies 16KB and 32KB could be optimal I-cache and D-cache size for NpBench application. As shown in Figure 3.6, the applications having control plane functions show more sensitivity on varying cache sizes. Also, it is found that CommBench and NpBench show similar trends in cache miss rates, for example, poor performance in data cache behavior.

In general, each application can be implemented with one PE having its L1 cache within the PE itself, and L2 cache of the network processor can be shared by several PEs. For reduction of L2 memory access latency, a few mechanisms are proposed [45][98].



(a) Instruction Cache Miss Rate (%) with varying cache size



(b) Data Cache Miss Rate (%)with varying cache size

Figure 3.6 Cache performance of network processor benchmarks

3.4.4 Available Instruction Level Parallelism

Instruction level parallelism of NpBench applications is investigated as a function of the inherent data dependencies and data flow constraints with limited number of functional units. Figure 3.7 shows that the available parallelism ranges from 2 to 9. As shown in Figure 3.7, NpBench control plane applications (7.61 on an average) have more ILP than NpBench data plane applications (4.80), and the available parallelism of CommBench (6.14) is in the middle. Security applications except for AES exhibit lower parallelism due to the need to perform encryption tasks. Since AES is a block cipher algorithm, it shows relatively higher ILP than other security applications. Even though the applications having control plane function have large amount of branch operations, they have more execution parallelism, which means there exists a room to improve performance of control plane processor with more parallel implementation. While most of NPs are implemented with several PEs to acquire packet level parallelism (PLP), they still need more parallelism with instruction level parallelism (ILP) and intra-packet parallelism (IPP) within the PEs or control plane processors.

3.4.5 Required Computation Capability per Packet

Some control plane and data plane workloads, from NpBench (e.g., WFQ, RED) and CommBench (e.g., DRR, FRAG), are used to get the required computational capabil-

ity (in terms of number of instructions) per packet. These experiments employ one million packets of data as each input. As shown in Figure 3.8, the required processing capability of control plane is estimated from 880 to 4,800 instructions per packet, while data plane is from 330 to 440 (2,660 for control plane and 380 for data plane on average). From the graph in Table 3.4, it is seen that control plane functions need larger processing capability, since their algorithms have higher complexity to meet sophisticated network services. For example, WFQ has to estimate each packet's finish-time and then classify the incoming packet into different queues, in order to maintain fairness and support QoS. This makes the algorithm more complex and the number of instructions larger. In contrast to that, FRAG performs relatively simple algorithm to split packets into multiple fragments, requiring less processing capability.

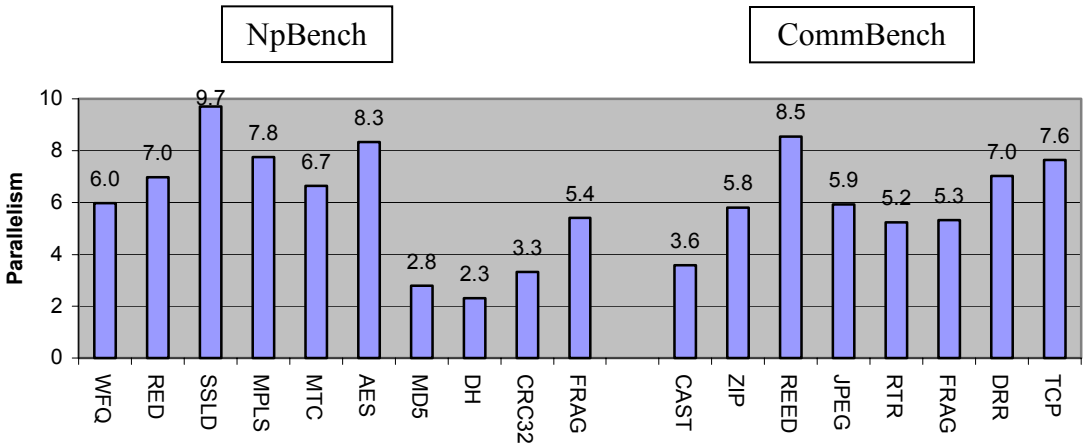


Figure 3.7 Available parallelism with ten function units

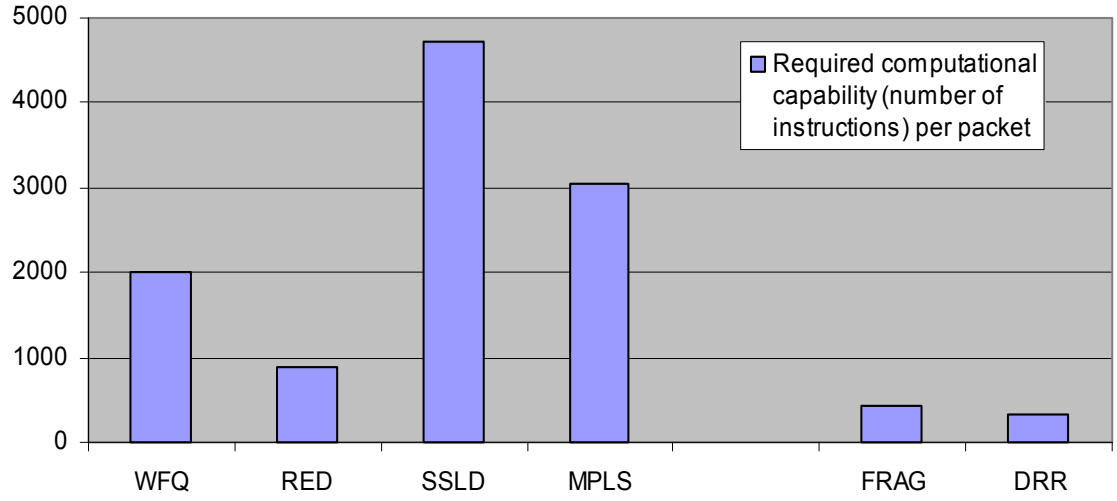


Figure 3.8 Required computational capability (in terms of number of instructions) per packet

As shown in Table 3.4, larger packet throughput is demanded for higher line rate. Assuming a stream of minimum-sized packets of 64 bytes and one clock frequency for executing one instruction, packet throughput of a 10 Gbps link is 19.5 million packets per second which means one packet is arrived every 51.2 nanosecond. Given a single processor of 1 GHz clock frequency, it can execute only 51 instructions per one packet time. Since a single processor is not enough to cope with wire speed and handle the workload of those applications, current trend of NPs is a single chip multi-processor. Not only more parallelism, but also new concept of instruction set architecture (ISA) with sophisticated programmability, should be considered to increase the number of instructions per cycle.

Table 3.4 Processing capability of single processor according to line rates and required processing capability of benchmarks

Line rate	Throughput (packets/s)	One packet time	Processor clock frequency	Allowable # of instructions per one packet time
1Gbps	1.95 M	512 ns	500 MHz	*256*
			1 GHz	*512*
10 Gbps	19.5 M	51.2 ns	500 MHz	*25*
			1 GHz	*51*
40 Gbps	78.12 M	12.8 ns	500 MHz	*6*
			1 GHz	*12*

3.5 ARCHITECTURAL IMPLICATIONS

Network processors can be used in various positions over the network such as edge, core or access network. Each application has different requirements and complexity. As a key solution to current network bottleneck, designing network processors could be a challenging work, for the reason that network environments keep changing and evolving. Since a variety of NP applications are used in the real field as shown in Table 3.1, network processors cannot be considered to be very focused application specific processors anymore. It has to process and perform various kinds of applications as does a general purpose processor, along with much higher throughput. When designing and evaluating a programmable network processor, especially for control plane processors, the workloads of NpBench can serve as an appropriate benchmark suite.

Since NPs would be used in routers over the network, most important things to be considered in designing NPs should be the processing capability without the slowdown of

required wire speed. Based on the characteristic analysis with NpBench, some issues should be reflected to design the network processor to accomplish demanding performance and throughput. The following issues are relevant:

- Based on the instruction mix used in executing benchmarks, new instruction sets should be considered to reduce the number of instructions per cycle and to accomplish higher throughput that can come up with the required number of instruction per packet.
- From the observation of cache behavior, data cache performance should be improved with novel cache structure for the network processor. When several PEs are integrated on a single chip, the problems including the shared memory problem should be solved for network processors.
- Due to variety of applications, packet level parallelism has been implemented with several processing engines. However, if large numbers of PEs are used, the processing time for each individual packet would be longer and utilization ratio could be deteriorated [2][70]. Based on the analysis of available parallelism, NP architectures still need more parallelism with instruction level parallelism (ILP) within the PEs or control plane processors. Intra-packet parallelism (IPP) should also be considered.
- Although most of NPs have been targeted for data plane applications, they also play a major role in the control plane. In fact, with the increased demand for complex processing, the boundaries between data plane and control plane have become blurred. To cope up with future data rate and complexity of the NP application, control plane

workloads should also be considered for designing the network processor, whether the NPs provide the function of control processor with an integrated core or externally via a host interface.

3.6 SUMMARY

As the network environment is rapidly changing, network interfaces demand highly intelligent traffic management in addition to the basic requirement of wire speed packet forwarding. Extracting representative applications and characterizing network workloads is essential for designing network processors and for evaluating their performance. Several vendors are releasing various network processors in order to handle these demands, but it is oriented for data plane functions to get more throughputs. Also, existing benchmark suites for the network processor primarily contain data plane workloads, which perform packet processing for a forwarding function.

In this chapter, a set of benchmarks, called NpBench targeted towards control plane workloads as well as data plane workloads are presented. The characteristics of NpBench workloads such as instruction mix, cache behavior, available parallelism and required processing capability per packet are presented and compared with CommBench. Also, analytical results including the architectural characteristics of the application having control plane functions, their implications to designing network processors and the significance of additional parallelism to perform NP applications at wire speed are discussed.

Chapter 4: Bottlenecks in Network Processor Applications

This chapter presents the bottleneck analysis of network processor applications. Bottleneck analysis is performed to see what is main bottleneck during the execution of control plane applications, how does the architectural factors affect the performance, and what kinds of architectural solutions are required to relieve the bottleneck.

4.1 Experimental Methodology

In order to study the sensitivity of each hardware resources towards the performance and the effectiveness of issue widths, Experiments on an out-of-order superscalar processor model which have variety of hardware resources are performed. Performance and power consumption are used as metrics, hence two different tools in this evaluation are utilized: The Simplescalar out-of-order simulator[17] is used for evaluating the performance of given architecture. The tool Wattch [16] is also used to estimate the power dissipation of given architecture when executing each application.

Several processor configurations are simulated on the different tools. For the experiments on the effectiveness of issue widths, various superscalar configurations ranging from 4-issue to 64-issue are simulated. Simplescalar configurations for 4 and 8-issue superscalar are explained in Table 4.1 and the wider superscalar configurations use proportionally larger resources.

For the bottleneck analysis, eight NP applications from the three NP benchmarks - CommBench, NetBench and NpBench - are chosen. Table 4.2 summarizes selected applications.

Table 4.1 Architectural configurations for the experiments

Hardware resorces	Issue width	
	4-issue	8-issue
Decode width	4	8
Issue width	4	8
Commit width	4	8
Integer ALU	4	8
Integer Multi/Div	4	8
FP ALU	1	1
FP Multi/Div	1	1
L1 I-cache	Size:32K, block size: 64, assoc:1	
L1 D-cache	Size:32K, block size:32, assoc:4	
L2 u-cache	Size:1,024K, block size: 64, assoc:4	

Table 4.2 Selected workloads: 8 NP applications and 3 multimedia kernels

Applications	Description
DRR	Deficit round robin scheduling
FRAG	Packet fragmentation application
REED	Reed-Solomon error correction scheme
WFQ	Weighted fair queuing
RED	Random early detection algorithm
SSLD	SSL(Secure sockets layer) dispatcher
MPLS	Multi-protocol layer switching
MTC	Media transcoding

4.2 EFFECTIVENESS OF WIDE ISSUE PROCESSORS

The most significant concern in designing network processors is maintaining the required throughput. Network processor workloads contain a large amount of instruction level parallelism [64]. Hence they possibly will be able to exploit wider and wider issue widths. For each NP application, various issue widths of superscalar architectures are applied in order to see the performance (IPC) variations; the result is shown in Figure 4.1. 4-issue superscalar is considered as a base configuration and the number of hardware resources is doubled according to each issue width. Most applications show better performance (IPC) with increasing issue widths, but some applications, such as FRAG, REED and SSLD, show early saturation at a small issue width. In order to get a high throughput, an aggressive increase of issue width can slightly help to improve the performance, but the cost and complexity of the hardware and the relatively small magnitude of improvement make wide issue not so desirable.

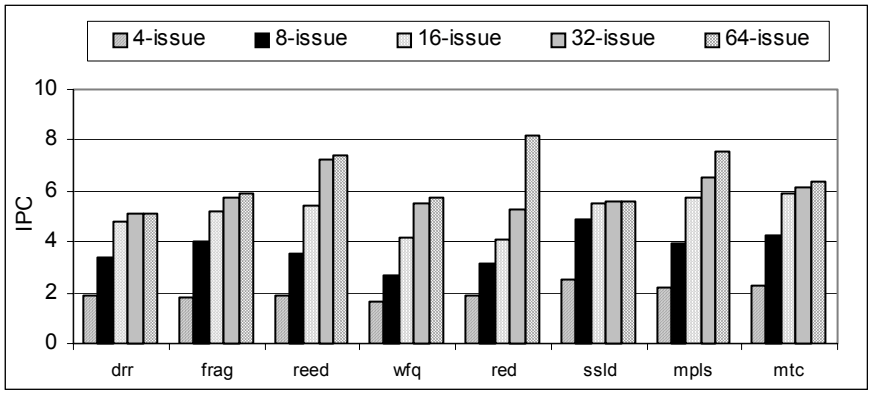
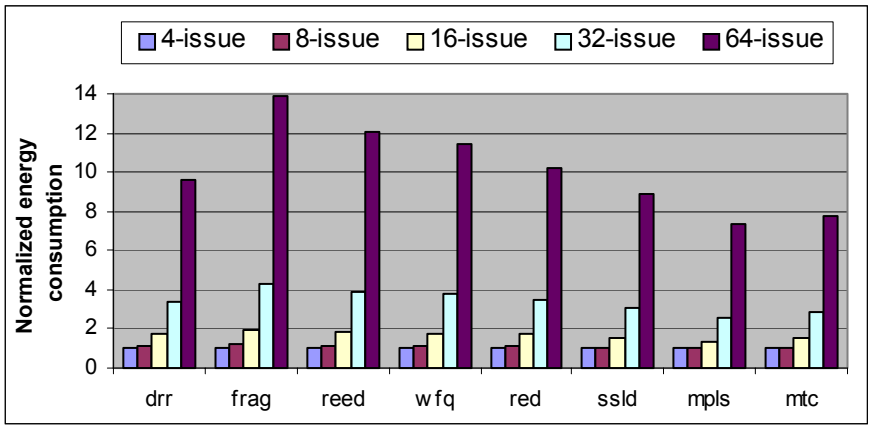


Figure 4.1 Performance impact of wide issue in NP applications

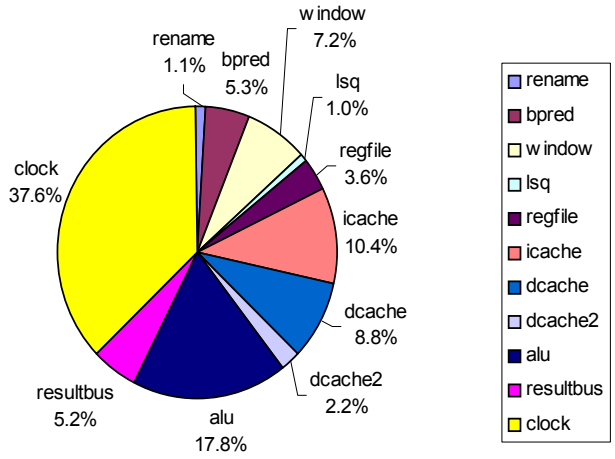
4.3 POWER CONSUMPTION OF WIDE ISSUE SUPERSCALARS

In order to understand the power consumption of wide superscalar processors, the Wattch [16] framework is used for experimentation. As shown in Figure 4.2, total energy consumption increases with increasing issue width. In this experiment, large amounts of power are consumed in instruction window wakeup/select, reorder buffer, and other scheduling related hardware modules. It is found that the total energy consumption of a 64-issue architecture is 10 times (on average) larger than that of 4-issue architecture. Figure 4.3 shows power distribution in dynamic execution of NP applications. In particular, the power consumption of the instruction window greatly increases with increasing issue widths.

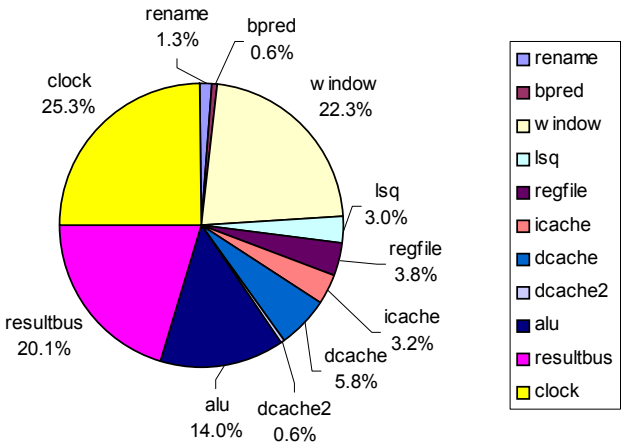


(Y axis: a normalized energy compared to the total energy of 4-issue superscalar architecture)

Figure 4.2 Energy consumption of wide issue Superscalar architectures for NP applications



(a) Power Breakdown of baseline architecture (4-issue)



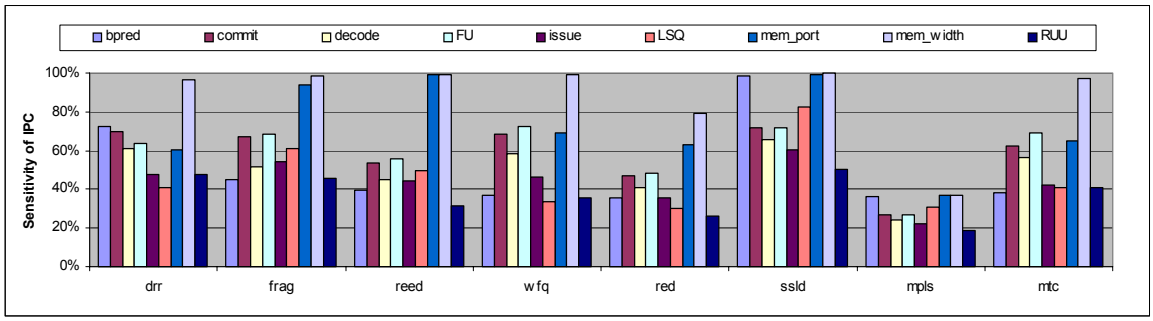
(b) Power Breakdown of baseline architecture (32-issue)

Figure 4.3 Power distribution in dynamic execution of NP applications

4.4 SENSITIVITY ANALYSIS

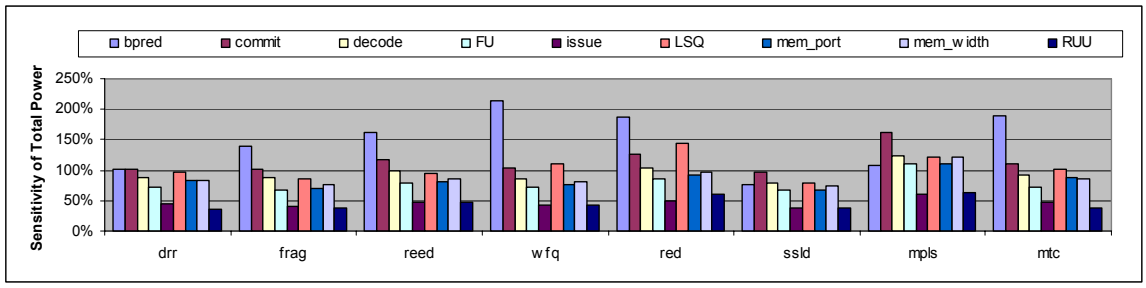
In order to investigate which hardware element is most influential to the throughput, sensitivity analysis for NP applications is performed. In this experiment, nine restricted hardware elements are used, including branch prediction, commit width, decode width, the number of functional units, issue width, load/store queue size, the number of memory ports, memory bus width and the register update unit.

Figure 4.4 presents the results of the sensitivity analysis for NP applications. For each NP application, the impact of restricting the resource is studied. In this analysis, the performance of a 32-issue machine is assumed as the maximum performance, since performance improvement is saturated at 32-issue based on the experiments. In each experiment, a single constraint is intentionally inserted into the maximum performance model. Each hardware element of the 4-issue superscalar is used as the corresponding constraint for the maximum performance model. From the experiment, the degree of impact, which indicates how the constraint affects the overall performance during dynamic execution, is investigated. The percentage value of each bar represents a normalized performance metric, which is the relative performance compared to the assumed baseline performance (100%).



(Y axis: IPC normalized with respect to 32-issue architecture)

(a) Loss of IPC when resources are restricted



(Y axis: Energy changes compared to baseline architecture)

(b) Changes (increase/decrease) in total energy consumption when resources are restricted

Figure 4.4 Sensitivity analysis with respect to the resource constraints in NP applications

For this sensitivity analysis, nine constraints, which are independent of each other, are applied: The 'bpred' bar shows the effect of branch misprediction compared to perfect prediction. The 'commit', 'decode' and 'issue' bar show the impact of the limited size of each resource. The 'FU' bar illustrates the impact of restricted functional units.

The ‘LSQ’ and ‘RUU’ bar show the effects of limited load/store queues and register update units, respectively. The ‘mem_port’ bar provides the sensitivity of the limited number of memory system ports available to the CPU, and the ‘mem_width’ bar represents the sensitivity of limited memory access bus width. From this analysis, it is found that the restriction of memory width has little impact on the overall performance in most NP applications, except for RED and MPLS. Branch misprediction has largely affected all NP applications, except for SSLD. This observation shows that branches are quite unpredictable in NP applications. As shown in Figure 4.4 (a), MPLS is the application that was most affected by all of constraints. The common bottlenecks across all NP applications are ‘LSQ’ and ‘RUU’, with RED and MPLS having the largest impact. Also, the ‘commit’, ‘decode’ and ‘issue’ width are medium-level bottlenecks in the overall performance for all NP applications.

Figure 4.4 (b) shows the sensitivity analysis of the total energy with respect to the resource constraints. It is interesting to note that branch misprediction leads to a large amount of additional energy dissipation, which is due to the increase in cycles due to the misprediction. The ‘commit’ has a similar effect as the ‘branch’ in some applications. Experimental results show that better performance (and hence fewer cycles) mean less energy consumption. All other constraints, except for the above two, show the proportional impact of the reduced resources in the power dissipation.

Table 4.3 shows the impact of inserted constraints on detailed resource elements in one representative application. The WFQ is selected for this experiment because WFQ shows a typical characteristic among the selected NP applications. When the ‘bpred’ is given as a constraint, the power dissipation of all resources (except for ‘LSQ’) is in-

creased in order to execute additional instructions, which compensate for misprediction penalties. The most affected resource is the register file in the WFQ experiment. The register file consumes twelve times more energy by restricting 'commit' width. This is because there are large amounts of access to the register file, which is due to the narrow commit width.

The 'issue' and 'RUU' make the largest impact on the power dissipation across all applications, which implies that large amounts of power is demanded for the related resources (e.g., instruction window) in large issue-width architecture. Figure 4.3 shows that the dynamic energy consumption of the instruction window is increased from 7.2% to 22.3% with larger issue architecture among the selected NP applications. It is assumed that aggressive clock-gating is employed and, therefore, power is scaled linearly with port or unit usage. Another assumption is that unused units dissipate 10% of their maximum power.

Table 4.3 Impact of resource constraints on energy distribution (WFQ)

Inserted Resource Constraints	**Energy Dissipation in each Hardware Elements**											***total***
	rename	**bpred**	**window**	**LSQ**	**regfile**	**icache**	**dcache**	**dcache2**	**alu**	**resultbus**	**clock**	
bpred	3.28	7.11	2.63	0.63	2.07	2.36	1.70	2.71	3.01	2.73	2.98	*2.14*
commit	0.90	1.46	1.00	0.33	12.23	1.48	0.71	1.47	1.43	1.00	1.46	*1.04*
decode	0.25	1.72	1.00	0.35	0.94	1.76	1.03	1.73	1.68	1.00	1.21	*0.86*
FU	1.25	1.38	1.01	0.32	1.22	1.07	0.91	1.38	0.38	0.76	0.65	*0.73*
issue	1.57	2.16	0.14	0.46	0.03	1.57	1.24	2.17	2.13	0.31	1.55	*0.43*
LSQ	2.18	2.96	1.12	0.38	2.20	1.82	1.59	2.97	2.91	1.32	1.62	*1.09*
mem_port	1.41	1.44	1.00	0.04	1.86	0.98	0.32	1.50	1.36	1.01	1.21	*0.78*
mem_width	1.10	1.01	1.00	0.27	0.95	0.83	0.71	1.01	1.01	1.00	1.13	*0.81*
RUU	0.92	2.83	0.09	0.28	2.32	2.25	1.50	2.84	2.80	0.34	1.47	*0.43*

4.5 SUMMARY

Generally, conditional operations and memory operations could be constraints for getting large parallelism and throughput. Based on previous research [64][65], most of NP applications use large amount of conditional operations irrespective of the amount of load and store usage. It is because they have to handle each packet by its priority and specific conditions to perform differentiated services, security checking and traffic management.

Based on the experiments in this chapter, an aggressive increase of issue width can help to slightly improve the performance in order to get a high throughput, but the cost and complexity of the hardware can be very high. For the energy perspective, total energy consumption more sharply increases with increasing issue width, when comparing performance perspective. The issue width is one of the bottlenecks for executing NP applications, but associated optimization techniques should be applied in order to reduce the hardware cost and to increase the utilization ratio of processing element.

The common bottlenecks across all NP applications are ‘load store queue’ and ‘register update unit’. Also, the ‘commit’, ‘decode’ and ‘issue’ width are medium-level bottlenecks in the overall performance for all NP applications. Branch misprediction leads to a large amount of additional energy dissipation, which is due to the increase in cycles due to the misprediction. The ‘issue’ and ‘register update unit’ make the largest impact on the power dissipation across all applications, which implies that large amounts of power is demanded for the related resources (e.g., instruction window) in large issue-width architecture.

The experiments in this chapter are based on dynamically scheduled architectures, so large amount of energy is consumed by a group of hardware resources (e.g., instruction window) for dynamic optimization. In general, static scheduled architecture requires relatively low energy consumption. Therefore, in the next chapter, NP applications are executed on statically scheduled architecture and performance and energy consumptions compared in order to see its effectiveness.

Chapter 5: Architecture with Statically Identified Parallelism

Most significant issues in modern processor design are related to exploiting parallelism to increase throughput and reducing power consumption. These issues are important in the design of Network processors also. Based on previous study [64], network processor workloads contain a large amount of instruction level parallelism and iterative execution of same algorithm. From this viewpoint, architectural characteristics of NP applications are similar to media applications. However, they are totally different from the data (packet) attribute aspects. While media applications show regularity in the data workloads and similarity between neighboring data, NP applications have irregular data workloads and distinct properties between adjacent data.

Based on similarity between media applications and NP applications, it is investigated whether the success of VLIW in the multimedia field can be applied to the network processor domain.

5.1 BACKGROUND

Contemporary computer and communication applications are multimedia-rich, involving significant amounts of audio and video compression, image processing, graphics, speech and character recognition and signal processing [12][32][41][42][63][66][67][83][91][97][99][105][106][107][108][109][110]. To handle these applications, many vendors have tried application-specific processors. Multimedia processors with VLIW archi-

tectures have been very popular. More recently, another application-specific processor, the network processor (NP), has been introduced to keep up with the high computation workloads and intelligent processing capability of network workloads [49][80].

Network processors handle packet data applications, control plane applications and payload applications [49][64][122]. Multimedia applications can actually be included in the network payload application category. Also, multimedia data is a dominant element of network bandwidth, and it might be a major reason of congestion problems, since the size of multimedia data sets tend to be large and differential services become widely applied [84].

Modern network applications and protocols demand intelligent and sophisticated processing over the network, which requires non-trivial computation capability. The processing requirements within network interfaces and routers are becoming more complex. The network processor on the physical port of a router should be able to process the modern workloads without slowing down line speed. Hence large parallel architecture is required for fast network processing, and relevant power issues are also considered in designing network processors [114].

Various kinds of network processors based on different architectural platforms are commercially available and academically studied [96]. Crowley, *et al*. [28] presented that simultaneous multithreading is best suited for some of network applications. Recent research and commercial products for network processors show the use of multithreading and vector-type array processing [24][74][80][122][123][124][125]. Although these implementations are targeted to meet the required performance, they have complexity in hardware and programming software, cost and power problems.

Current popular media processors – Texas Instruments' C6x [113] and TriMedia's TM-1300 [47][97] - rely on the simpler hardware of VLIW processors in order to minimize the cost and power of ILP implementation [48]. Some network processors, such as Cisco Toaster [22] and Agere's PayloadPlus [3][71], are partially applying static scheduling/VLIW to their architectures. The basic architecture of Toaster is a systolic array architecture, and PayloadPlus is employing VLIW to one of several components. However, it is still not widely thought that VLIW is the way to go for network processors. For recent research related to VLIW, Rao, *et al*. [92] evaluates compiler support for network processing.

In this chapter, it is investigated whether the success of VLIW in the multimedia field can be applied to the network processor domain as a processing element of a parallel architecture for NP as shown in Figure 5.1. Generally, the processing element should be a simple and low power architecture. Therefore, the VLIW architecture can be a good candidate for the processing element of NP parallel architecture from VLSI design aspects. This premise is analyzed through the comparison between network processor workloads and multimedia workloads in terms of performance (speedup) and power consumption. Network processor applications are quite different from multimedia and DSP applications in functionality, but both applications have large data (packet) level parallelism. However, network processor applications have not-so-regular parallel characteristics compared to multimedia applications. It is also analyzed how these different characteristics in parallelism affect the performance.

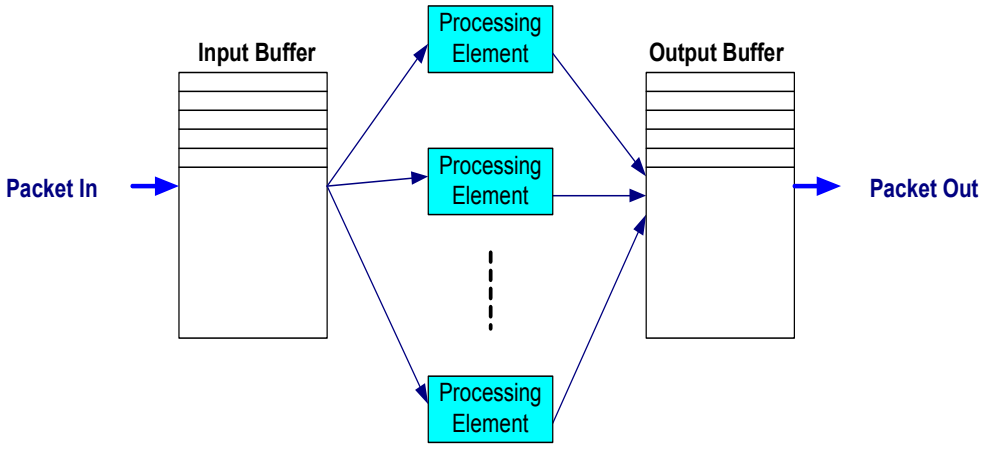


Figure 5.1 Conceptual structure of parallel architecture for network processors

5.2 WORKLOADS

Eight NP applications are selected from the three NP benchmarks [64][76][122], and three multimedia kernels are also included for the experiments. Table 5.1 summarizes selected applications.

Based on previous research [64][122], it is observed that most of NP applications use large amount of conditional operations irrespective of the amount of load and store usage. It is because they have to handle each packet by its priority and some specific conditions to perform differentiated services, security checking and traffic management.

In general, conditional operations and memory operations typically consume a significant part of the total execution cycles, and also they restrict performing the parallelism.

Table 5.1 Selected workloads: 8 NP applications and 3 multimedia kernels

Applications	Description
DRR	Deficit round robin scheduling
FRAG	Packet fragmentation application
REED	Reed-Solomon error correction scheme
WFQ	Weighted fair queuing
RED	Random early detection algorithm
SSLD	SSL(Secure sockets layer) dispatcher
MPLS	Multi-protocol layer switching
MTC	Media transcoding
MM	Matrix multiplication
ADPCM	Adaptive Differential Pulse Modulation
FFT	Fast fourier transform

Figure 5.2 shows the flow of WFQ [11][14][30][103]. WFQ is a good example to investigate its data flow because it has many characteristics of exploiting QoS (Quality-of-Services) [126], and it is also a solution for congestion problem of the network interfaces.

The WFQ has two main loops – Loop2 handles all queue operations, and during Loop1, round number calculations for next Loop2 are performed based on results of Loop2 iterations. Therefore, as shown in Figure 5.3, NP applications (e.g. WFQ) have both parallelizable operations and an amount of serial operations, while most of multimedia applications have large amount of parallelizable operations. However, there is still a possibility to improve parallelism for WFQ, since parallel processing parts are dominant of the total execution. The flow analysis of WFQ evidences that statically scheduled ar-

chitecture, which shows good performance in multimedia areas, can be a candidate of a processing element for network processors.

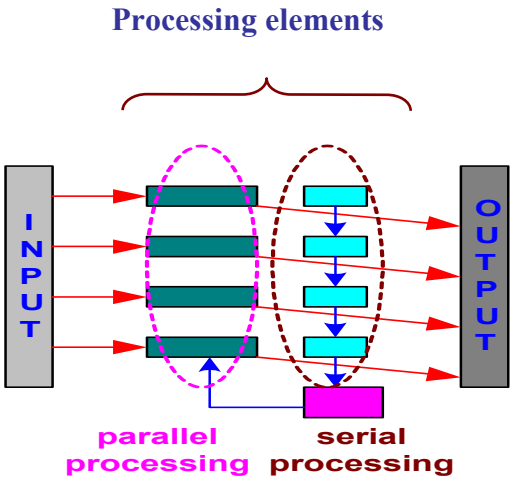
```
Loop1: (active)
   Loop2: (for all queues)
      1. Empty queue checking
      2. Finish time calculation
      3. Calculation of min/max value of finish time
   EndLoop2

   4. Round number calculation for next Loop2
EndLoop1
```

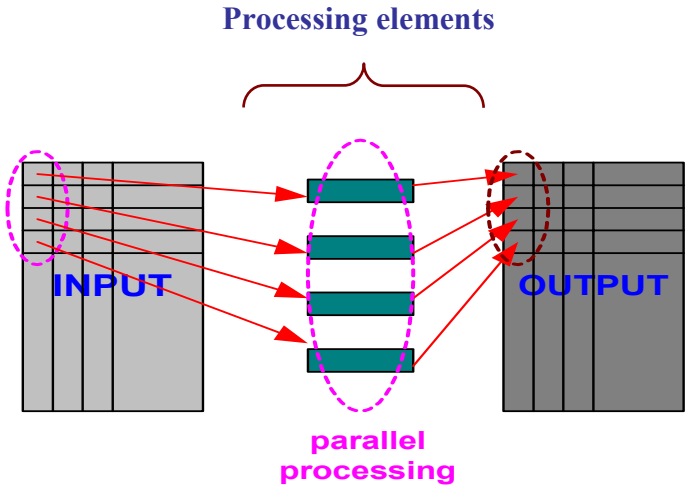
Figure 5.2 Main flow of the WFQ application

5.3 EXPERIMENTAL FRAMEWORK

In order to study the effectiveness of static scheduling for NP applications, experiments are performed on a VLIW architecture model. However, any particular statically scheduled architecture is not advocated in this experiment. VLIW paradigm is simply applied as a vehicle to investigate the feasibility of static scheduling for NP applications. Performance (speedup) and energy consumption are used as metrics, hence power simulators are also needed. The Trimaran [113] tool is used as a framework for static scheduled architecture simulation. The tool PowerImpact [68] is also used to estimate power consumption of VLIW architecture. For the simulation, 8-issue Trimaran configuration is applied as shown in Table 5.2. The VLIW configuration (4-1-2-1) is based on instruction distribution presented in chapter 3 and also in [64].



(a) NP application (e.g. WFQ)



(b) Multimedia application (e.g. FFT)

Figure 5.3 Data flow of NP and multimedia workloads

In the static scheduling simulation using Trimaran, three different region formations techniques - basicblock, hyperblock and superblock – are used in order to see the

effectiveness of aggressive compiler optimization techniques. Basicblock scheduling has a limited scope of exploiting ILP, and each basicblock has 4-5 interdependent instructions on average. Hyperblock and superblock are a kind of extended basicblock for scheduling in which groups of basicblocks are scheduled as a single unit.

PowerImpact is designed on the Impact tool [20]. These tools are capable of breaking down the power consumption with respect to the various units of the processor, and hence analysis of the power consumption of various processor units is analyzed in detail.

Table 5.2 Architectural configurations for the VLIW experiments

VLIW	Trimaran-4121 configurations
Integer	4
Floating Point	1
Memory	2
Branch	1
L1 I-cache	Size:32K, block size: 64, associativity:1
L1 D-cache	Size:32K, block size:32, associativity:4
L2 u-cache	Size:1,024K, block size: 64, associativity:4

5.4 Performance and Power Characteristics of NP and Multimedia Workloads on Static Scheduled Architecture

In this section, the performance of NP applications is compared to multimedia applications on static scheduled architectures (VLIW).

5.4.1 Performance Metric in VLIW

Total execution cycles is used as a performance metric rather than IPC (Instruction per Cycle), when comparing the performance impact between different techniques with respect to one application. It is because some aggressive compiler optimizations increase the total number of dynamic instructions. Even though the optimization techniques introduce much higher parallelism, IPC comparison based on different number of instructions are not reasonable within one application. For example, as shown in Table 5.3, superblock simulation of MTC has highest IPC among three block formations, but its actual execution cycle is not the highest. Also, in the static scheduling experiments, it is seen that some applications, such as WFQ and RED, has better performance when executing the application with superblock optimization, while MPLS, SSLD and MTC have better results with hyperblock.

5.4.2 Performance Characteristics

In VLIW architecture, the compiler plays a major role in finding parallelism, decreasing dependencies among instructions and exploiting other optimization techniques in static mode. For more aggressive optimization, several types of region formations have been used in the compilation stage. Table 5.4 shows the static code size of each region formation in VLIW optimization. Hyperblock and superblock optimization has a much larger code size than basicblock optimization, as shown in Table 5.4, since these optimizations use several algorithms, such as tail duplication, node splitting and loop peeling, to exploit larger parallelism. These algorithms make the code size larger during the optimi-

zation process, but total execution cycles become smaller due to the aggressive parallelism from the optimization.

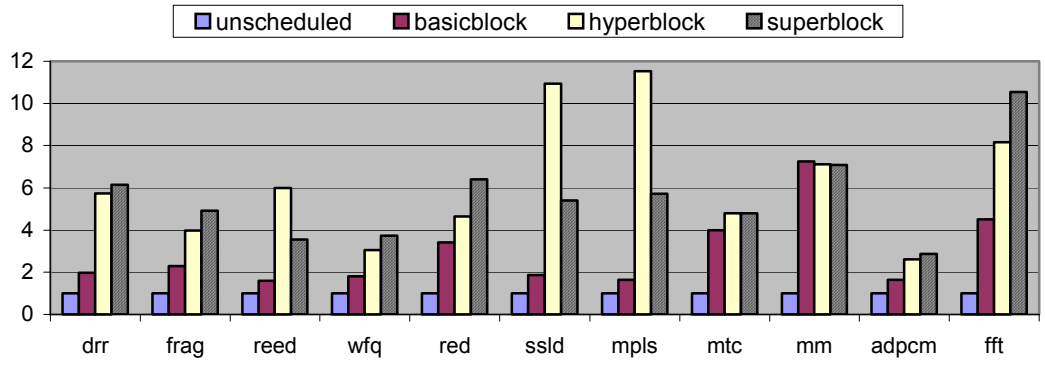
Table 5.3 Performance characteristics for selected NP applications and multimedia kernels on VLIW

Benchmarks/Architectures			Total icount	**Total cycles**
DRR	Trimaran-4121	basicblock	299,861,954	255,950,125
		hyperblock	246,863,079	87,865,261
		superblock	**242,616,913**	**81,910,851**
FRAG	Trimaran-4121	basicblock	46,701,519	29,800,918
		hyperblock	46,415,391	17,113,605
		superblock	**47,315,391**	**13,813,605**
REED	Trimaran-4121	basicblock	791,093,230	664,971,864
		hyperblock	**764,712,720**	**175,726,369**
		superblock	1,085,431,602	297,961,729
WFQ	Trimaran-4121	basicblock	21,840,219	20,205,926
		hyperblock	30,549,203	11,965,495
		superblock	**31,366,939**	**9,716,299**
RED	Trimaran-4121	basicblock	8,619,963	5,134,718
		hyperblock	11,560,637	3,770,468
		superblock	**10,901,051**	**2,734,384**
SSLD	Trimaran-4121	basicblock	41,540,697	32,216,891
		hyperblock	**22,430,498**	**5,480,132**
		superblock	38,446,616	11,115,653
MPLS	Trimaran-4121	basicblock	47,130,280	37,635,484
		hyperblock	**26,018,784**	**5,346,058**
		superblock	43,394,820	10,761,597
MTC	Trimaran-4121	basicblock	423,951,206	249,137,276
		hyperblock	**411,585,030**	**207,468,467**
		superblock	413,819,818	207,553,368
MM	Trimaran-4121	**basicblock**	**4,978,201**	**1,731,232**
		hyperblock	5,705,563	1,762,825
		superblock	5,724,763	1,770,025
ADPCM	Trimaran-4121	basicblock	10,259,845	7,425,995
		hyperblock	11,812,295	4,682,874
		superblock	**11,480,874**	**4,253,279**
FFT	Trimaran-4121	basicblock	32,932,660	16,776,649
		hyperblock	16,128,966	9,281,398
		superblock	**18,706,169**	**7,176,384**

Table 5.4 Static code size of different region formation techniques

Benchmarks	Static code size of different region formations		
	basicblock	hyperblock	superblock
DRR	204	1,170	1,073
FRAG	89	152	152
REED	148	915	847
WFQ	353	2,258	1,875
RED	358	858	1,332
SSLD	280	1,551	1,778
MPLS	263	926	976
MTC	988	3,160	3,221
MM	166	621	573
ADPCM	172	469	491
FFT	580	2,074	2,268

The most important exploiting of parallelism can be done by employing instruction scheduling which is for assigning instructions into fixed functional units in VLIW architecture. Figure 5.4 shows the performance comparison between the scheduled and the unscheduled VLIW experiments. For a more intuitive comparison, normalized speedup is used for comparison to total execution cycles of the unscheduled VLIW.



** Y axis: a normalized speedup to the total execution cycles of unscheduled VLIW*

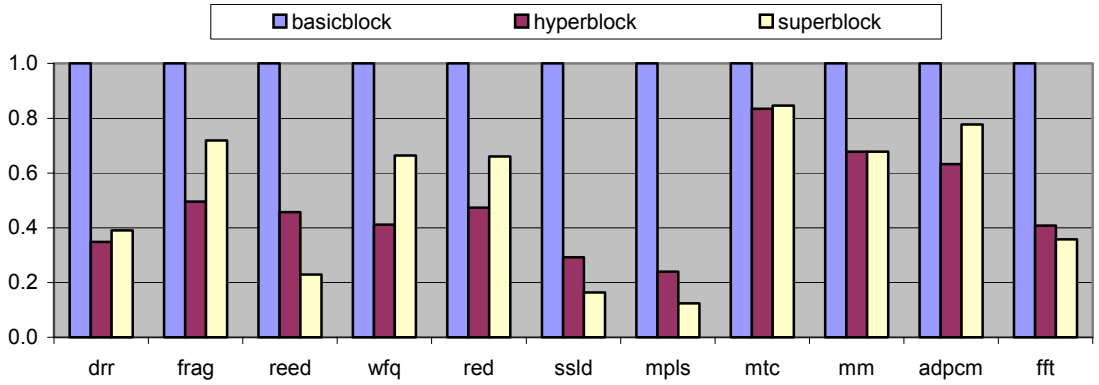
Figure 5.4 Performance characteristic on VLIW

Most of NP applications show relatively little improvement from basicblock optimization compared to multimedia workloads. However, more aggressive optimization techniques give comparable improvements to NP workloads. It is because hyperblock and superblock optimizations reduce the impact of conditional operations, which is a main factor to decrease the parallelism of NP workloads. MPLS and SSLD show tremendous impact from hyperblock optimization. Another interesting observation is that multimedia applications show comparable performance in basicblock optimization without using hyperblock and superblock optimization except ADPCM. In the case of ADPCM, its impact from scheduling is little, because each sample is computed using previous sample values making it to contain very little parallelism. Based on this experiment, it can be found that the static scheduled architecture can be applied to network processor domain with more aggressive optimization techniques.

5.4.3 Power Characteristics of NP and Multimedia Workloads on VLIW

Power consumption of NP and multimedia applications on VLIW architectures is investigated to see the power impact from using different block formation techniques. Figure 5.5 shows normalized power consumption to basicblock formation. From Figure 5.5, it cannot be observed any regular patterns of power consumption among different block optimization techniques in both applications, but NP applications have more energy impact than multimedia applications from aggressively optimized block formation. It is because larger parallelism requires more power consumption to exploit more hardware

units. Based on the power experiments, similar power characteristics are expected between both workloads when more optimized parallelism is applied to NP applications.



** Y axis: a normalized power consumption to basicblock formation*

Figure 5.5 Power consumption on VLIW

5.4.4 Required Parallelism for NP Workloads

Assuming a stream of minimum-sized packets of 64 bytes and given a single processor of 1 GHz clock frequency for the router, this single processor can execute only 512 instructions per one packet time over 1Gbps (~OC-24) network line. As shown in Table 5.5, required number of parallelism can be obtained with 1~10 parallel implementation of a unity ILP machine [64]. If OC-192 and OC-768 routers are considered, the requirements will be much higher as shown in Table 5.5. Based on this study, it is seen that the parallel implementation costs can be reduced by 50%~75% from a single VLIW-4121. Even though more works on tuning issue-width and accelerating processing elements are required, static scheduled architecture can be a candidate for a processing ele-

ment when designing network processors as a processing element of a parallel architecture. As the architectural technology is being evolved, higher speed processors also help to reduce the cost of parallel implementation.

Table 5.5 Required parallelism for NP workloads
(* is indicates control plane applications)

Workload	Required number of processing elements (with a unity ILP processor)			Parallelism of a VLIW-4121	How many processing elements (VLIW-4121) are required?		
	(~OC-24)	(OC-192)	(OC-768)		(~OC-24)	(OC-192)	(OC-768)
DRR	1	7	28	4.04	1	2	7
FRAG	1	9	38	3.23	1	3	10
WFQ*	4	39	168	2.07	1	10	42
RED*	2	17	74	3.22	1	5	19
SSLD*	10	91	394	8.62	3	23	98
MPLS*	6	59	255	5.71	2	15	63

5.5 SUMMARY

In this chapter, it is investigated whether the success of VLIW in the multimedia field can be applied to the network processor domain as a processing element of a parallel architecture for NP. This premise is analyzed through the comparison between network processor workloads and multimedia workloads in terms of performance (speedup) and power consumption. Network processor applications are quite different from multimedia and DSP applications in functionality, but both applications have large data (packet) level parallelism. However, network processor applications have not-so-regular parallel characteristics compared to multimedia applications. It is also analyzed how these different characteristics in parallelism affect the performance. Experimental results show that NP

applications need more aggressive optimization techniques in static scheduled architecture, while media applications can get large parallelism with simple basicblock optimizations. With the characteristics of large packet-level parallelism, experimental analysis supports static scheduling as an applicable paradigm for network processor applications with lower hardware complexity and lower power dissipation.

Based on the experiments, it can be concluded that control plane workloads need large-scale parallel implementations. Generally, network processors can be used in various node positions with different scales, such as core routers (10 Gbps), edge routers (2.5 Gbps) and access routers (1 Gbps). Large scale routers may need a grid style architecture to meet the required throughput, but it is hard to apply such a complex and expensive architecture to small scale routers. Therefore, a simple and low priced NP architecture is required for small scale routers. In the following chapters, the feasibility of using a hardware accelerator for specific heavily used operations from the application will be studied. Careful analysis of the workload is conducted to identify appropriate candidates for hardware acceleration.

Chapter 6: Hardware Acceleration Techniques for Control-plane Workloads in Network Processors

This chapter describes hardware acceleration techniques for control plane workloads in network processors. Three types of hardware acceleration modules presented in each section. This chapter is organized as follows: Section 6.1 describes congestion control workloads and a hardware acceleration technique. Section 6.2 provides media transcoding workloads and an array-style hardware acceleration model. Section 6.3 presents lookup table related workloads in control plane and a hardware acceleration technique using partitioned lookup table. Finally, summary of this chapter is presented in Section 6.4.

6.1 Hardware Acceleration for Congestion Control Applications

Complex network protocols and various network services require significant processing capability for modern intelligent network applications. One of the significant features in modern networks is differentiated service. Along with differentiated service, rapidly changing network environments cause congestion problems. In this section, characteristics of representative congestion control applications such as scheduling and queue management algorithms are analyzed, and application-specific acceleration techniques are proposed using PLP (Packet Level Parallelism). From PLP perspective, a hardware acceleration model is proposed based on detailed analysis of congestion control applica-

tions. In order to get higher throughput, large number of processing elements and a parallel comparator are designed. Such hardware accelerators provide large parallelism proportional to the number of processing elements added.

6.1.1 Motivation

As the bottleneck of communication networks is moving to the network nodes from channel bandwidth problem, it is required to have flexible processing capability for network processors in order to support several emerging network applications and their heavy processing workloads [49][80].

One of the significant features in modern network workloads is differentiated service. This differentiated service requires a capability to process computation-intensive workloads in the network node. Also, as network contents tend to be large and various kinds of client devices are introduced, new network applications such as media transcoding [41][103] should be performed in the network nodes. These rapidly changing network environments also bring about congestion problems. While the existing research and products [27][37][51][52][76][122] have been mostly focused on handling packet processing, no previous research analyzes these workloads from the architectural perspectives.

NP workloads have large data (packet) level parallelism, since they iteratively use same algorithm to input packets [64]. In order to extract and exploit the parallelism for designing network processors, it is extremely important to identify appropriate benchmarks for efficient design and evaluation of any processor. Among three NP benchmarks

[64][76][122], NpBench [76] provides a categorized NP control-plane workloads. Control plane workloads are just emerging and evolving in current network environments, and they perform congestion control, flow management, higher-level protocols and other control tasks.

In general, congestion occurs at a router when incoming packets arrive at a rate faster than the rate that the router can switch them to an outgoing link as shown in Figure 6.1. The two representative algorithms for congestion control are the scheduling and the queue management algorithm [15][39]. The scheduling algorithm determines which packet to be sent next and is used primarily to manage the allocation of bandwidth among flows (e.g., weighted fair queuing (WFQ) [11][30][103]). According to the IETF (Internet Engineering Task Force) recommendation [15][39], the default mechanism for managing queue lengths in FIFO queues is the Random Early Detection (RED) algorithm.

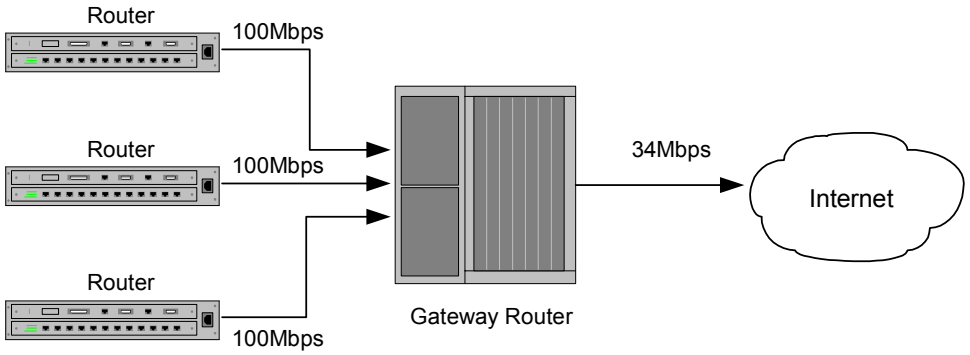


Figure 6.1 Router system

For the low-delay and low-jitter services (e.g. premium services), the network processor on the physical port of a router should be able to process these congestion control workloads without slowing down the line speed. Assuming a stream of minimum-sized packets of 64 bytes and given a single processor of 1 GHz clock frequency for the router, this single processor can execute only 512 instructions per one packet time over 1Gbps (~OC-24) network line. As shown in Table 6.1, required number of parallelism can be obtained with 2~4-way parallel implementation of a unity ILP machine [64]. In the case of OC-192 and OC-768 routers, the requirements will be much higher as shown in Table 6.1.

Table 6.1 Required parallelism for congestion control workloads

Workloads	Required number of instructions per packet	Required parallelism (with a unity ILP 1 GHz processor)		
		1Gbps (~OC-24)	10Gbps (OC-192)	40Gbps (OC-768)
WFQ	2,005	4	40	167
RED	881	2	17	74

In this section, characteristics of two representative congestion control applications are analyzed, and application-specific acceleration techniques are proposed using PLP concepts. For PLP implementation, a hardware acceleration model is proposed based on detailed analysis of congestion control applications.

The rest of this section describes workload characterization of congestion control applications and performance analysis of proposed acceleration model.

6.1.2 Workload Characterization

In order to design coprocessors that relieve congestion, a detailed understanding of the algorithm itself including its dataflow, essential operations and its inherent bottleneck is required. In this section, two representative applications for congestion control - WFQ and RED are analyzed. The WFQ [14][104] is a queue-scheduling algorithm to serve packets in order of their finish-times considering the weight on connections. Various lengths of packets from incoming traffic are classified into different queues, which can be used for differentiated service. The Random Early Detection (RED) [15][39] is an active queue management algorithm for routers. In contrast to the traditional queue management algorithm, which drops packets only when the buffer is full, the RED algorithm probabilistically drops arriving packets before coming into the queue. The decision of whether or not to drop an incoming packet is based on the estimation of the average queue size.

As shown in Figure 5.3, NP applications (e.g. WFQ) contain several parallelizable operations. Particularly, two congestion solutions show many comparison operations and conditional operations to exploit fair scheduling and active queue management.

6.1.2.1 Experimental framework

Workload characterization and performance analysis are performed with the Simplescalar Tool Set version 3.0 for the PISA architecture [17]. Program performance is analyzed on the detailed timing simulator. In order to measure more accurate timing requirements, proposed acceleration model is coded using VHDL. Synopsys Design Vision

[105] is used for the synthesis of the VHDL model, in which a cell-based methodology is used to target the VHDL models to a 0.18-micron technology.

6.1.2.2 Kernel Characteristics

Instruction mix of the congestion control applications is investigated. Table 6.2 shows the dynamic instruction distribution during execution. From this workload distribution, it can be observed that ALU operations occupy a significant share of the total instruction mix (21% ~ 40%). Branch operations (branch, jump and call) are heavily used in the congestion applications (16% ~ 30%) for finding fair conditions and active queue management, compared to crypto applications (7% ~ 10%) - AES, MD5 and DH. These conditional operations are a main factor to decrease the parallel performance of congestion control applications.

Table 6.2 Instruction distribution of congestion control applications and crypto applications

App.	alu	shift	logic	branch	Load	store	msc.
WFQ	20.6	16.9	0.0	29.2	16.2	7.9	9.3
RED	39.7	7.2	0.0	15.3	23.6	10.9	3.0
AES	10.5	18.4	26.9	7.0	29.4	7.7	0.1
MD5	45.1	13.0	20.5	7.5	7.1	6.7	0.2
DH	24.0	12.0	10.9	9.7	27.0	11.3	5.1

6.1.2.3 Bottleneck Analysis

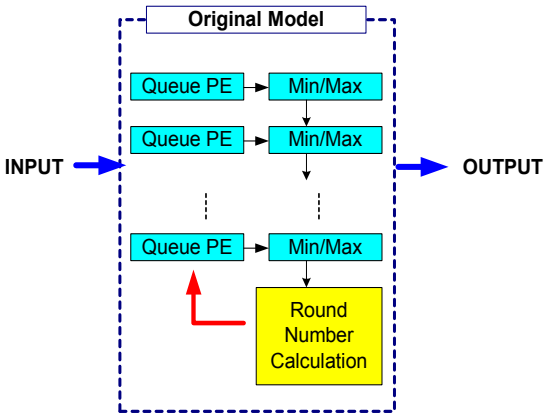
In order to investigate which hardware element impacts performance the most, bottleneck analysis is performed for congestion applications. For this bottleneck analysis, nine constraints, which are independent of each other, are applied - branch prediction, commit width, decode width, the number of functional units, issue width, load/store queue size, the number of memory ports, memory bus width and the register update unit.

From this analysis, it is observed that the restriction of memory width has less impact on the overall performance in WFQ than RED. Branch misprediction has largely affected both applications. This observation shows that branches are quite unpredictable in these congestion applications. The common bottlenecks across both applications are 'bpred', 'issue' width, 'LSQ' and 'RUU'.

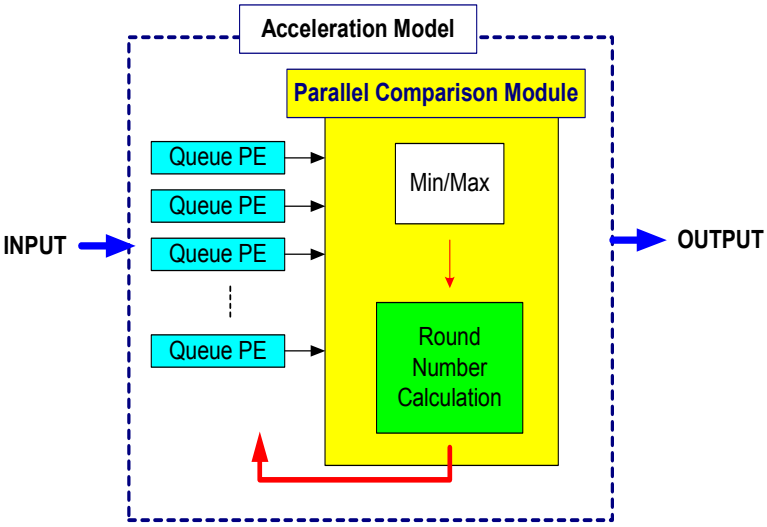
In the following section, it is focused on relaxing bottleneck effects from 'issue' width using the hardware acceleration model, in order to maximize congestion control application's inherent parallelism.

6.1.3 Acceleration Model

Along with the instruction level parallelism, packet level parallelism should be applied to congestion control application to acquire the required parallelism, even though they have serial data flows as shown in Figure 6.2 (a). The input packets are processed in the queue processing elements (PEs) for the fair queuing, and min/max calculation and round number calculations are performed based on the finish time of each processing elements.



(a) Original architecture



(b) Proposed accelerator architecture

Figure 6.2 Hardware acceleration model for WFQ

In order to exploit packet level parallelism, min/max calculation module is decoupled from the queue processing flow, and merged into the large parallel comparison module with round number calculation model. Also, the queue processing elements are configured in parallel for exploiting packet level parallelism as shown in Figure 6.2 (b).

Given N_P packets and N_Q queues, the total execution cycle of Figure 6.2 (a) can be expressed as follows:

$$\{(t_Q + t_M) \cdot N_Q + t_R\} \cdot \frac{N_P}{N_Q}$$

where t_Q is queue processing cycles, t_M is min/max calculation cycles, and t_R is round number calculation cycles. Under the same condition, the total execution cycle of the proposed hardware acceleration model is:

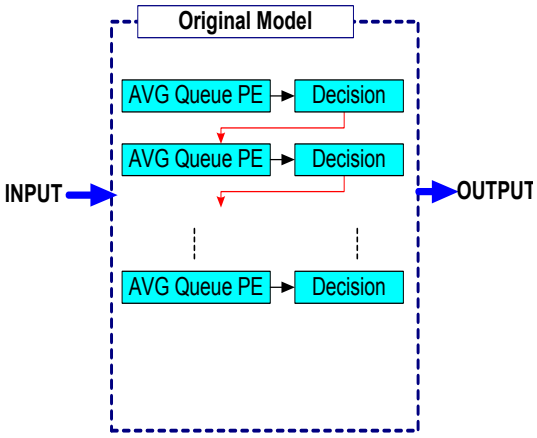
$$\{t_Q + t_{MR}\} \cdot \frac{N_P}{N_Q}$$

where t_{MR} is min/max calculation cycles for all queue processing elements plus round number calculation cycles (t_R). Therefore, the expected performance improvement is:

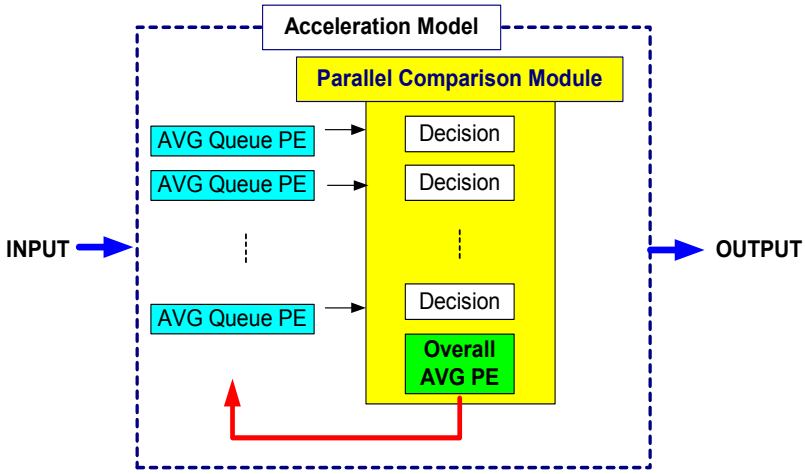
$$\frac{t_Q \cdot N_Q + t_M \cdot N_Q + t_R - t_Q}{t_{MR}}$$

From the above equation, only N_Q dominantly affects the performance improvement. Hence, the speedup from proposed hardware acceleration module would be proportional to the number of queue processing elements and inversely proportional to t_{MR}. If same acceleration mechanism is applied to RED as shown in Figure 6.3, the average

queue calculation module can be modeled as front-end PLP elements, and decoupled decision module and overall average queue size calculation module can be configured at the back-end as a parallel comparison module. Similarly, the speedup from proposed hardware acceleration module would also be proportional to the number of front-end PEs.



(a) Original architecture



(b) Proposed accelerator architecture

Figure 6.3 Hardware acceleration model for RED

6.1.4 Experimental Results and Performance Evaluation

The proposed hardware acceleration model, as explained in section 6.1.3, is simulated for performance evaluation. In this experiment, 'min/max calculation time' (see Figure 6.2) for large input sets is assumed to be approximately 10 cycles in the WFQ. It is based on generic 2-bit comparator delay, which is 3 XNOR gates delay. And, the 2-bit comparisons should be performed over '$\log_2^{(\text{number of input sets})}+1$' stages. So, if 64-queue processing elements is applied and 8-bit comparison is needed, total delay of the critical path could be 3 XNOR gates delay (3x170ps) x 7 stages x 3 (8bit comparison). Given 1GHz clock frequency and assumed 65 ps as one gate delay, it takes about 10 cycles.

In order to measure more accurate timing requirements, parallel comparator is modeled using VHDL and synthesized using Synopsys synthesis tools [105]. Cell-based methodology is used to target the VHDL models to a 0.18-micron technology (HT018.db). From the synthesis, critical-path delay of the parallel comparator is 4,166.27 ps, which is 5 cycles if 1 GHz clock frequency is applied.

As shown in Figure 6.2, actual decision for sending a packet happens at the end of queue processing (Queue PE), and the next round of queue processing requires a round number from current round queue processing. From the experiment using Simplescalar, round number calculation time is approximately 142 cycles, and min/max calculation time at the back-end is assumed as 5 cycles. These delays are constant regardless of the number of front-end PEs, when $t_M < t_R$. As shown in Figure 6.4, speedup from proposed hardware acceleration module is proportional to the number of front-end PEs.

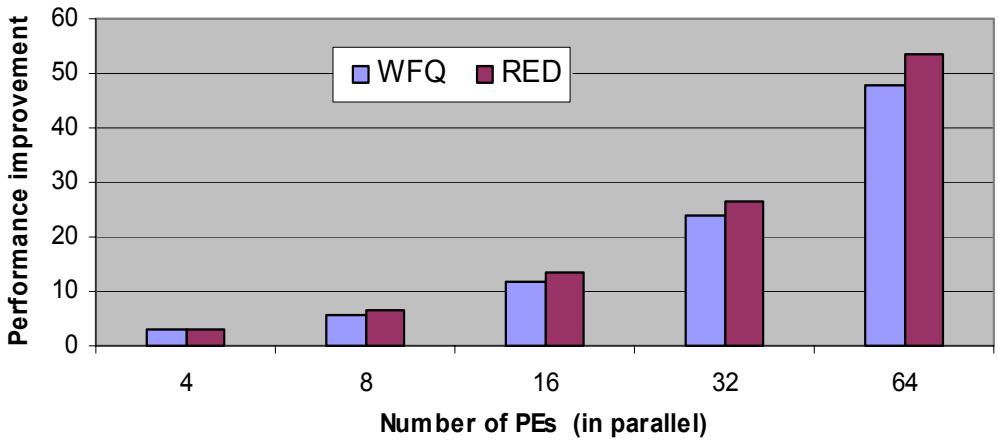


Figure 6.4 Performance evaluation of hardware acceleration model for congestion control applications

The performance comparison is based on the execution time of 4-issue Simplescalar architecture. The RED shows more improvement, since its ratio of front-end execution cycle to back-end execution cycle is larger than that in the WFQ. In the case of RED, the number of PEs should be 4, 8 and 32 for OC-24, OC-192 and OC-768 to avoid slowing down the line speed. It is observed that the RED shows more improvement, since its ratio of front-end execution cycle to back-end execution cycle is larger than that in the WFQ.

In order to get large parallelism in congestion control applications, it is required to carefully decouple the dataflow and split into the front-end PLP module and the back-end hardware acceleration module. This decoupling and defining a hardware acceleration module could be done by the thoroughly analysis of the applications. Since large number of processing elements should be applied for congestion control applications, implemen-

tation issues such as power, processing element utilization ratio, interconnection delay can not be ignored.

6.2 HARDWARE ACCELERATION FOR MEDIA TRANSCODING APPLICATIONS

In this section, characteristics of media transcoding applications are analyzed, as the second category of control plane applications for network processors, and application-specific acceleration techniques are proposed to exploit data (packet) level. Array-style processing module is considered as a candidate for accelerating data conversion module. Such hardware accelerators provide large parallelism proportional to the number of array processing elements added.

6.2.1 Motivation

As network contents tend to be large and various kinds of client devices are introduced, new network applications such as media transcoding [46][100] should be performed in the network nodes. As shown in Figure 6.5, the network bandwidths of content server and client device has a big difference, since content servers are directly linked to large scale networks (e.g., enterprise network) and client device is related to small scale networks (e.g., home networks, mobile networks). Also, the hardware resources for displaying and processing data are very limited in client device aspects, particularly for mobile devices [34][35]. If high resolution data is to be sent to client devices from content server, it is not economical to send original high resolution data. It will take longer to

send the data over the small bandwidth network, and processing time in the client device also will take long time. In addition to that, it can cause congestion problems in the node. Therefore media transcoding technology should be essentially applied to intermediate node between content servers and client devices. Most of data are related to regular media data, so it would be well matched with array-style architecture to handle data transcoding. For PLP implementation using the array architecture, hardware acceleration models are proposed based on detailed analyses of media transcoding applications.

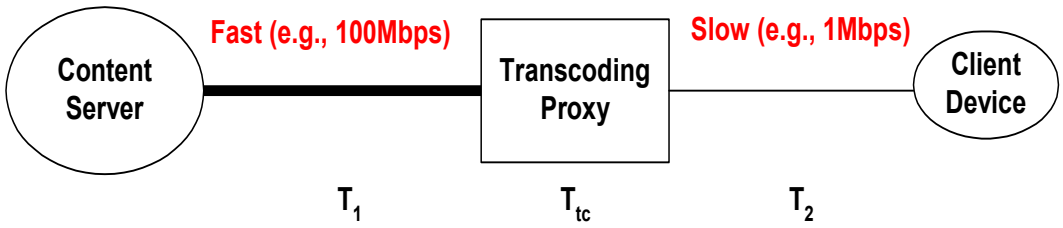


Figure 6.5 Media transcoding

The rest of this section describes workload characterization of media transcoding applications and performance analysis of proposed acceleration model.

6.2.2 Workload Characterization

Media Transcoding [46][100] is a process in which a data object in one representation is converted into another representation. In order to accommodate the wide spectrum of client capabilities and associated network speeds, the media data need to be

modified along the dimensions, fidelity, and resolution. It is not an efficient way to send a bigger data to low-resource devices through the slow network line. As shown in Figure 6.5, transcoding proxy will handle this conversion work, so network bandwidth will be efficiently used and it helps to avoid the congestion.

6.2.2.1 Experimental Framework

Workload characterization and performance analysis are performed with the Simplescalar Tool Set version 3.0 for the PISA architecture [17]. Program performance is analyzed on the detailed timing simulator. Several processor configurations are simulated for bottleneck analysis. Synopsys synthesis tool [105], *design vision*, is also used for timing verification of proposed hardware acceleration modules.

6.2.2.2 Kernel Characteristics

Dynamic instruction mix of the control plane applications is investigated. Table 6.3 shows the dynamic instruction distribution during execution. From this workload distribution, it can be observed that ALU operations occupy a significant share of the total instruction mix (33.9%). Memory operations (load and store) are heavily used for fetching data from the memory and storing the result to the memory. If systolic-style array processors are applied to this application, the memory fetching and storing workloads can be significantly reduced.

Table 6.3 Instruction distribution of media transcoding applications

App.	Alu	shift	logic	branch	load	store	misc.
MTC	33.9	2.6	10.5	11.2	21.7	18.1	1.9

6.2.2.3 Bottleneck Analysis

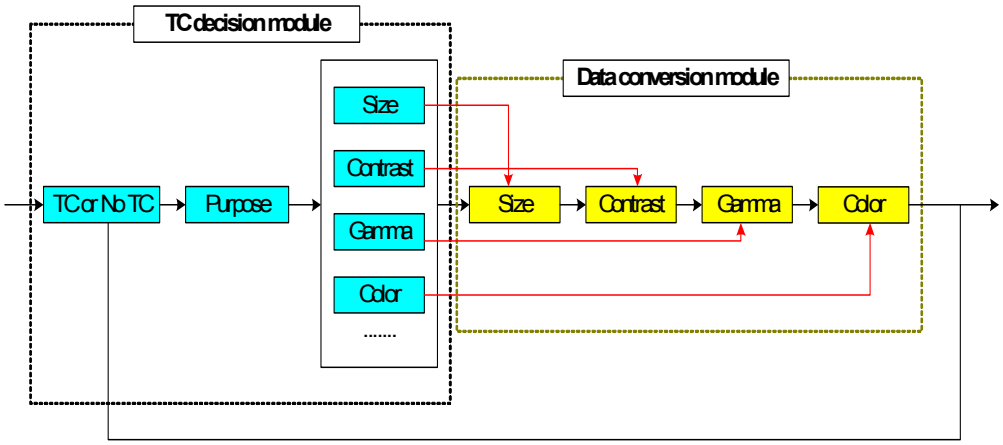
For the bottleneck analysis of media transcoding, nine constraints as section 6.1.2.3, which are independent of each other, are applied. Based on this analysis, it is found that the restriction of 'memory_width' has less impact on the overall performance. Branch misprediction has more impact on this application. The bottlenecks which strongly influenced the performance are 'bpred', 'issue width', 'LSQ' and 'RUU'. In the following section, it is focused on relieving bottleneck effects from 'issue' width and 'LSQ' using parallel hardware acceleration model, in order to maximize media transcoding application's inherent parallelism.

6.2.3 Acceleration Model

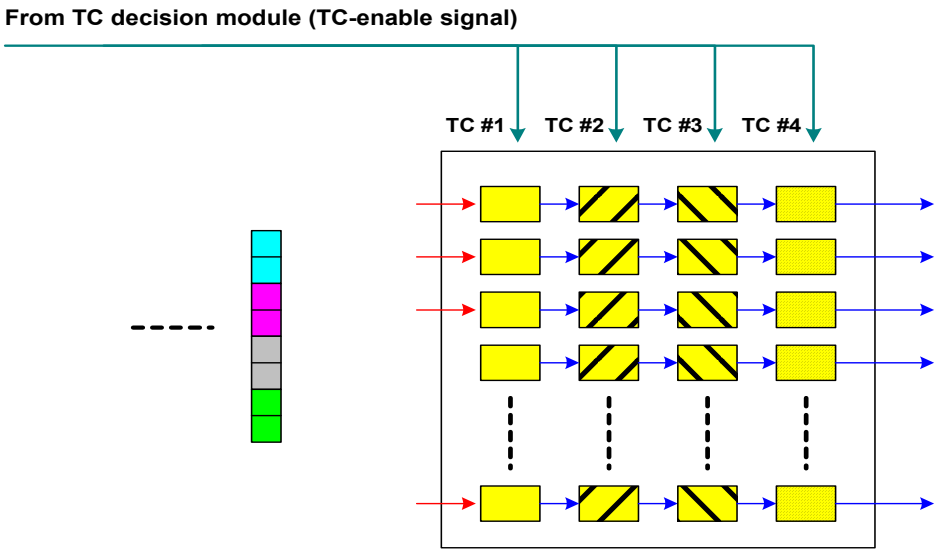
The procedure of media transcoding consists of transcoding decision unit and pipeline-styled regular processing units as shown in Figure 6.6 (a). This decision module is very critical in the functioning of the workload. The decision module handles which module is enabled and how many modules are enabled. Data conversion modules include several different kinds of functionalities including size conversion, color conversion, contrast control and gamma correction. Once each processing unit is enabled, the input data

is processed in one processing unit and the processed data is forwarded to the next processing unit. Most of data to be processed in the transcoding proxy are related to multimedia applications, so each processing unit requires regular and fine-grained processing element. Therefore, the array-style processor would be a good candidate for accelerating media transcoding as shown in Figure 6.6 (b).

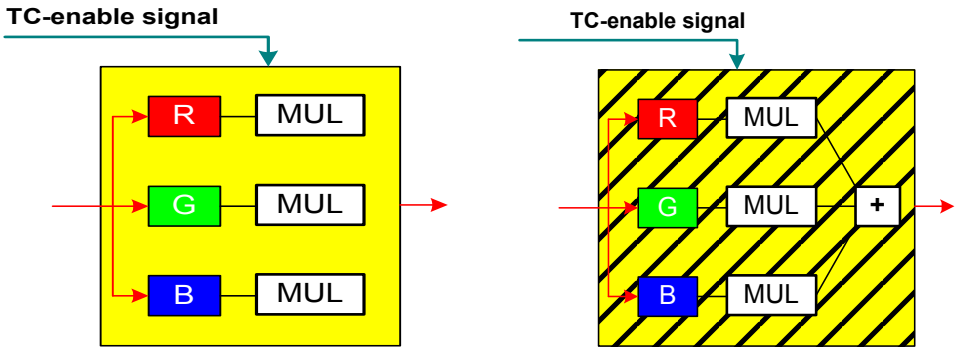
Based on experiments using 4-issue Simplescalar, data conversion module takes 89% of the total execution cycles. If a 128x128 image is used as an input to the data conversion module, average execution cycles for an image takes 58,800 cycles when four transcoding modules are enabled. As shown in Figure 6.6 (c), a processing unit consists of simple arithmetic elements including multipliers for red, green and blue color processing, or a combination of three multipliers and one adder.



(a) Original architecture



(b) Proposed accelerating architecture



(c) Processing units

Figure 6.6 Hardware acceleration model for media transcoding

6.2.4 Experimental Results and Performance Evaluation

Each processing unit consists of 3 multipliers and one adder as shown in Figure 6.6 (c). If array multiplier is applied, the critical path of the multiplier is 14 adder-delays. High-speed multipliers like Wallace multiplier can reduce the critical path to 7 adder-delays. In order to measure timing requirements, processing unit is modeled using VHDL and the model is synthesized using 0.18-micron technology (HT018.db). From the synthesis results, its critical-path delay of a processing unit is 7,784.96 ps, which is 8 cycles if 1 GHz clock frequency is applied.

If a 128x128 image is considered as an input to the data conversion module (without accelerator), average execution cycles for an image takes 58,800 cycles when four transcoding modules are enabled. If 4 parallel processing units are applied for each transcoding module (4x4 array architecture), it takes 32,768 cycles, which is 1.8x speedup. If 64 parallel implementation (4x64 array architecture) is used, only 2,048 cycles is needed for transcoding (28.7x speedup). Figure 6.7 shows the speedup of proposed acceleration module in terms of the number of PEs.

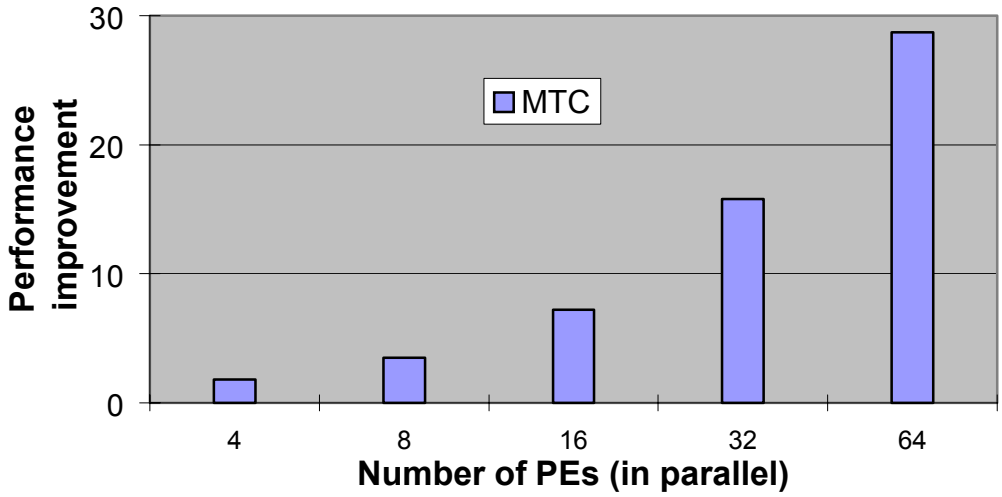


Figure 6.7 Performance evaluation of hardware acceleration model (media transcoding)

6.3 HARDWARE ACCELERATION FOR LUT-RELATED APPLICATIONS

In this section, characteristics of lookup-table-related applications for network processors are analyzed, and application-specific acceleration techniques are proposed to exploit data (packet) level parallelism in these applications. It is focused on accelerating lookup table searching module using the modified partitioned lookup table mechanism. Such hardware acceleration technique provides large parallelism proportional to the number of LUT partitions.

6.3.1 Motivation

As e-commerce applications are widely used in secured environments, content-based switching mechanism [7][100] between server and client clusters, which is a model to reduce the computational loads of heavy authentication, have emerged.

SSLD [100][103] is one example of content-based switching mechanism in the server and client environments. When SSLD runs over TCP connection, the SSLD maintains the session ID information during authentication process. When reconnecting to the same server, a client can reuse the session so that computational load of authentication which is computationally heavy processing, can be reduced. In order to find previously used session information and update newly used session information, lookup table (LUT) manipulations are applied.

MPLS [9][56][126] is a forwarding technology, which does away with the lookup of bulky IP headers and uses short labels for forwarding at the edge of the MPLS domain.

MPLS also use lookup table for finding a forwarding equivalent class of a given destination address.

Both SSLD and MPLS have lookup table searching and updating module in each procedure as show in Figure 6.8. Lookup table searching is a major bottleneck in the LUT-related applications. Based on experiments using 4-issue Simplescalar, LUT searching takes 77% ~ 86% of the total execution cycles in MPLS and SSLD. In this section, hardware acceleration models of LUT searching functionality are proposed.

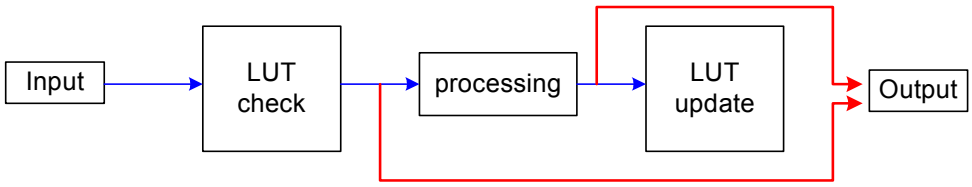


Figure 6.8 Data flow of LUT related applications

The rest of this section describes workload characterization of lookup table related applications and performance analysis of proposed hardware acceleration model.

6.3.2 Workload Characterization

Architectural characteristics of LUT-related applications are analyzed. Bottleneck analysis is also performed to see what is main bottleneck for executing these applications.

6.3.2.1 Experimental Framework

Workload characterization and performance analysis are performed with the Simplescalar Tool Set version 3.0 for the PISA architecture [17]. Program performance is investigated on the detailed timing simulator. Several processor configurations are simulated for bottleneck analysis. The tool *CACTI* [121] is used for timing verification of memory module for lookup table.

6.3.2.2 Kernel Characteristics

Dynamic instruction mix of the SSLD and MPLS applications are investigated. Table 6.4 shows the dynamic instruction distribution during execution. From this workload distribution, it can be observed that ALU operations occupy a significant share of the total instruction mix (35.8% ~ 57.0%). Branch operations are heavily used for comparison of unique ID during LUT searching. If hardware module for parallel searching is applied, actual execution time consumed by comparison operations can be reduced.

Table 6.4 Instruction distribution of MPLS and SSLD applications

App.	Alu	shift	logic	branch	load	store	misc.
MPLS	35.8	11.4	8.6	22.0	16.1	4.9	1.3
SSLD	57.0	0.0	0.0	28.3	14.4	0.2	0.0

6.3.2.3 Bottleneck analysis

For the bottleneck analysis of LUT-related applications, nine constraints as in section 6.1.2.3, which are independent of each other, are applied. Based on this analysis, MPLS is restricted by most of the constraints, while SSLD is mainly restricted by 'RUU' and 'issue width'. It is interesting to note that all hardware resources are evenly used in the MPLS application.

6.3.3 Acceleration Model

Lookup table searching is a major bottleneck in the LUT-related applications. In this study, the focus is on accelerating lookup table searching module. Akhbarizadeh, *et al*. [6][101] proposed partitioned lookup table mechanism for IP packet forwarding, which is applied for IPv4 or IPv6 [29]. As shown in Figure 6.9, they use parallel GT (Great Than) comparison block for each partitioned lookup table (PLUT) in order to find the longest prefix matching IP address and output port among partitioned lookup tables. They also use TCAM (Ternary Content Addressable Memory) module [43][113] for lookup table.

For PLP implementations of SSLD and MPLS, partitioned lookup mechanism from [101] is modified. Lookup table is partitioned into N (power of 2) small lookup tables. Each PLUT contains a subset of data for lookup. Lookup table of MPLS has desti-

nation IP addresses and FEC ID, while SSLD contains destination IP addresses and session ID in lookup table.

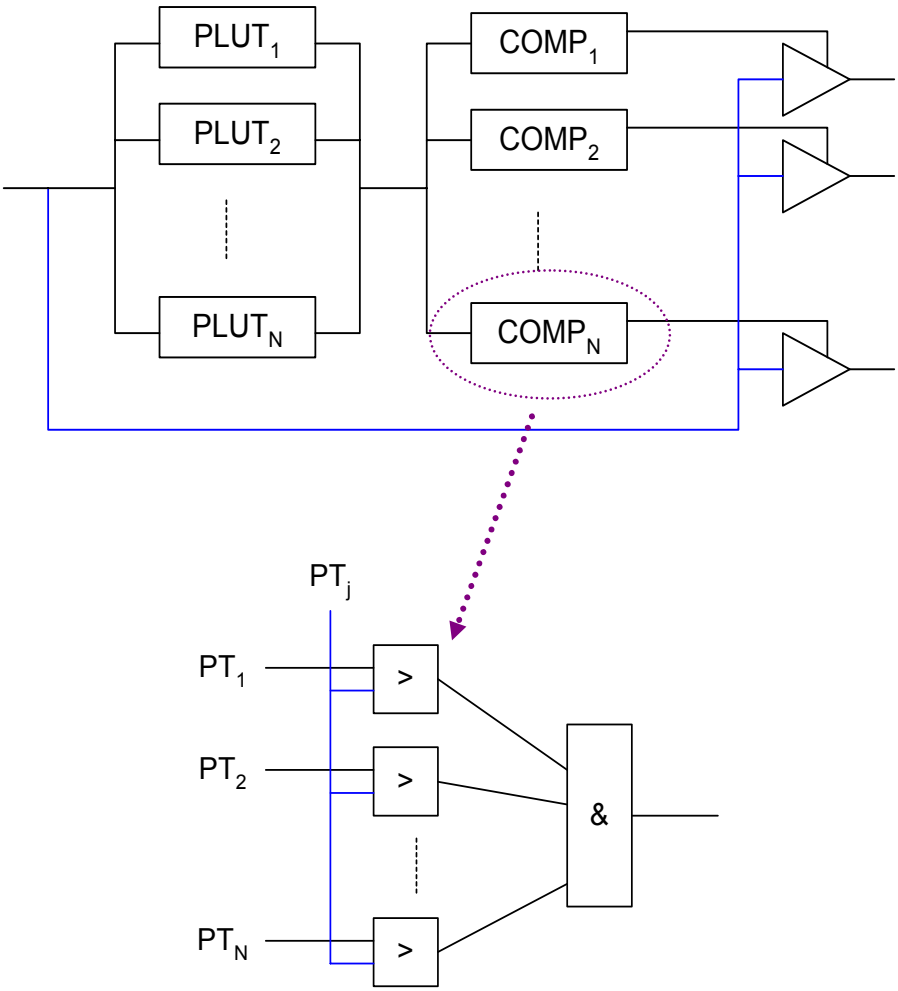


Figure 6.9 Partitioned LUT [6]

Figure 6.10 presents proposed partitioned lookup table mechanism. When an IP packet is received, its destination IP address field is broadcasted to all the PLUTs. The partitioned lookup table units are checking if the received destination exists in each PLUT. If it exists in a PLUT, associated data (FEC ID or session ID) will be controlled by Hit/Miss control signal whether the data is transferred to next stage or not. The PLUT is similar to the behavior of cache. In this approach, applying TCAM module for lookup table is not considered, because it still costs a lot, dissipates more power and takes more area compared to SRAM. Typically one bit cell of SRAM requires 4-6 transistors, while one bitcell of TCAM requires 11-15 transistors [43]. Several companies including Siber-Core Technologies [113] produce 2M – 18M TCAM. In the case of 2M TCAMs, it is capable of single and multi-field classification in as little as 10 ns. Currently TCAM is relatively expensive to use, but it can be expected to be much faster, denser and cheaper products in the near future [43].

6.3.4 Experimental Results and Performance Evaluation

Using the tool *CACTI* [57][121], average access time of the RAM-based lookup tables is compared, when applying different lookup table sizes.

In the case of lookup table for SSLD, it consists of 32 bit IP address and 256 bits for session id. From the CACTI simulation, the access time of this LUT module (256 Kbytes, 4K entries) is 2.48 ns under the 0.18 micron technology. If the lookup table size is 64 Kbytes (1K entries), the access time can be reduced to 1.65 ns. Therefore it can be

expected to get speedup in searching LUT if the partitioned LUTs can be executed in parallel for SSLD or MPLS. In the case of MPLS, 8 bytes of block size is needed for FEC ID. Hence, estimated access time is 1.31 ns for a 32 Kbyte LUT (4K entries) and 1.05 ns for an 8 Kbyte LUT (1K entries) respectively.

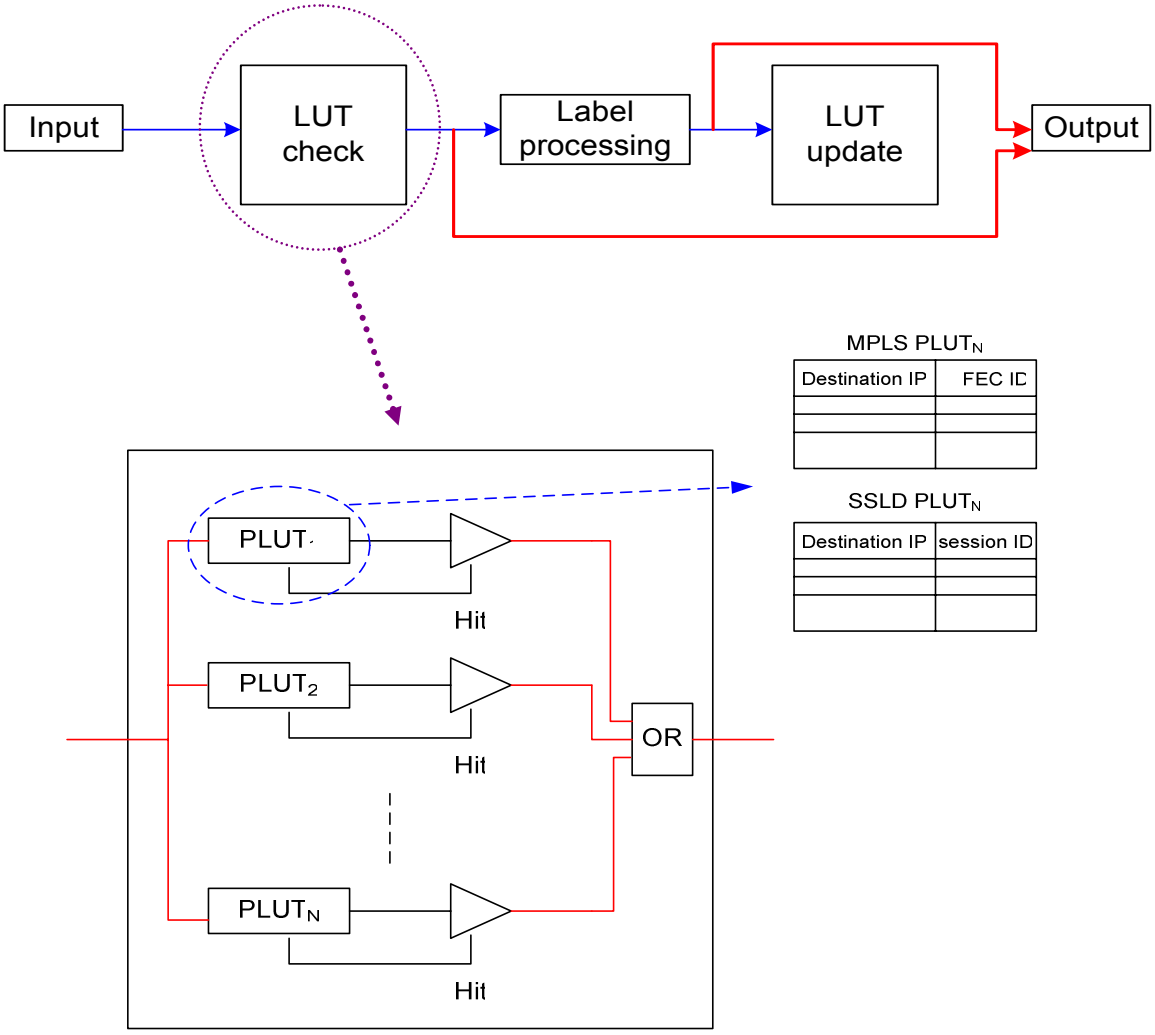


Figure 6.10 Hardware acceleration model for LUT checking module

Based on the simulation, total critical-path delay of LUT searching module is same as access time of a lookup table memory module. As shown in Figure 6.11, speedup from proposed acceleration module is 178x ~ 262x for MPLS and 163x ~ 362x for SSLD, when compared to execution cycle without acceleration module.

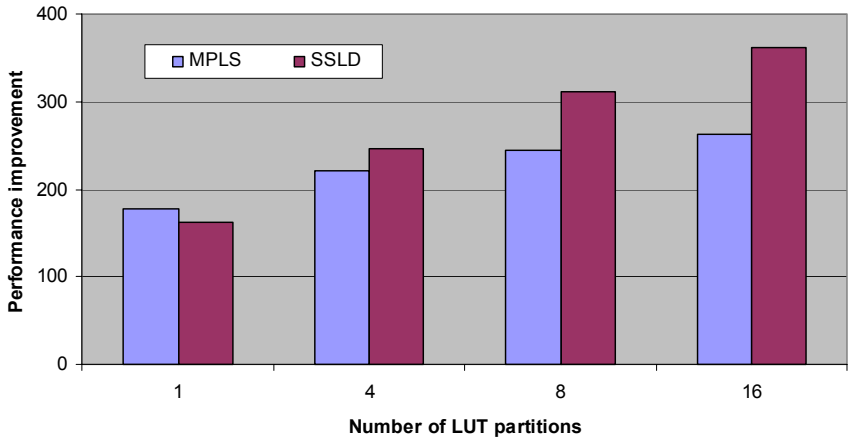


Figure 6.11 Performance evaluation of hardware acceleration model (LUT related application)

6.4 SUMMARY

In section 6.1, characteristics of network congestion control applications are analyzed, and application-specific acceleration techniques are proposed to exploit data (packet) level parallelism in these applications. Two representative algorithms for congestion control are scheduling and queue management algorithms. From the PLP perspective, a hardware acceleration model is proposed based on detailed analysis of congestion

control applications. In order to get higher throughput, large number of processing elements and a parallel comparator are designed. Such hardware accelerators provide benefits proportional to the number of processing elements added.

In section 6.2, characteristics of media transcoding applications for network processors are analyzed, and application-specific acceleration techniques are proposed to exploit data level parallelism. Most of the data to be processed in the transcoding proxy are related to multimedia applications, so each processing unit has regular and fine-grained processing element. Therefore, an array-style processor is used as a candidate for accelerating media transcoding.

Lookup table searching is a major bottleneck in the LUT-related applications. In order to reduce this bottleneck, in section 6.3, it is focused on accelerating lookup table searching module using the modified partitioned lookup table mechanism. Parallel comparator (for equality checking) is also used to find exactly matched session ID (for SSLD) or FEC (forwarding equivalence class) ID (for MPLS) among partitioned LUTs.

Current network processors handle specific application with each processing element for accelerating packet processing. This is because there is no common solution for network processing workloads and many companies want to use their own design solution to avoid any costs for licensing fee and to reduce the time-to-market. There is no research on finding common architectural characteristics on network processor workloads, while several architectural solutions are commonly used for multimedia workloads. The processor architecture for network nodes has been changed from the software implementation on general purpose processor to ASICs and network processors. As the performance of general purpose processor rapidly increases, it is required to consider going back

to use general purpose processors for network nodes, but more application-specific extension modules are required to commonly used for network processor applications.

Chapter 7: Instruction Set Extensions for Efficient Network Processing

One of the significant features in modern networks is differentiated service. Along with differentiated service, rapidly changing network environments result in congestion problems. In this chapter, application-specific acceleration technique is proposed using instruction set extensions for congestion control applications. Based on investigation, other control plane applications, except for media transcoding, have little opportunities to come up with new instruction set extensions. Multimedia extensions have already been applied to general purpose processors such as MMX.

As the performance and clock frequency of general purpose processor rapidly increases, one can consider going back to use general purpose processors for network nodes. However, as in multimedia extensions, it is essential to extract instruction set extensions for network processor applications.

7.1 MOTIVATION

To design coprocessors that relieve congestion, one requires a detailed understanding of the algorithm itself including its dataflow, essential operations and its inherent bottleneck. The WFQ [30][104] is a queue-scheduling algorithm to serve packets in the order of their finish-times considering the weight on connections. The Random Early Detection (RED) [15] is an active queue management algorithm for routers. In contrast to the traditional queue management algorithm, which drops packets only when the buffer is

full, the RED algorithm drops arriving packets probabilistically before coming into the queue. These two congestion solutions show many comparison operations and conditional operations to exploit fair scheduling and active queue management. In general, conditional operations and comparison operations reduce the parallelism in dynamic execution. If a few instructions associated with these operations can be combined into one instruction, total number of instructions can be reduced in dynamic execution.

7.2 INSTRUCTION SET EXTENSIONS FOR CONGESTION CONTROL APPLICATIONS

Based on analysis of congestion control applications, highly parallel implementation for packet (data) level parallelism (PLP/DLP) is needed to get high throughput as the network line speed become higher. In the case of OC-768 (40Gbps), 74 ~ 167 parallel implementation of a unity ILP machine is required. Even though a simple parallel architecture is modelled, its hardware cost and power consumption cannot be ignored. In order to reduce these costs, the performance improvement of a single processing element is an important factor in designing parallel architecture. In this dissertation, instruction extensions are proposed by defining new instruction sets to increase pereformance within a single processing element.

Table 7.1 shows new instructions to support fast execution of congestion control applications. These architectural extensions are created by detailed kernel analysis in source code and instruction level. Frequently used instructions are combining into one instruction. Basically, two groups of instructions are defined: conditional operations and multiplication-add / multiplication-sub. New conditional instructions including CNDGT,

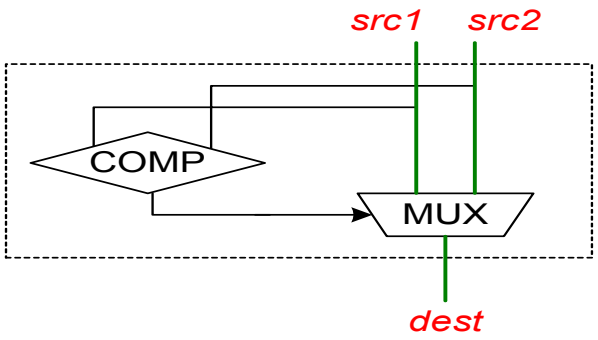
CNDLT, CNDEQ, CNDNE, CNDEZ and CNDNZ, are frequently used in congestion control applications to find fair conditions and active queue management. Based on the experimental results, congestion control workloads have many computational operations during dynamic execution. Therefore, two ALU operations are also combined as one instruction such as MADD1, MADD2, MSUB1 and MSUB2. These instructions can reduce the execution cycles in ALU. This kind of multiply-add instructions are already available in many general purpose architectures.

Table 7.1 A extensions for congestion control applications

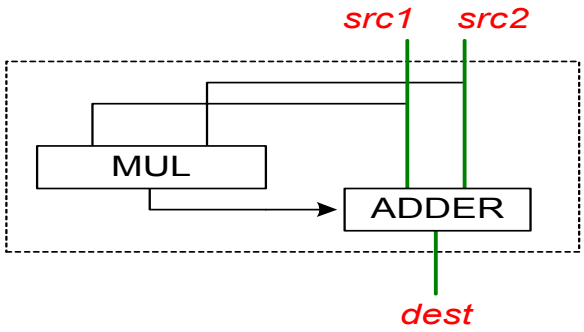
Instruction		**Description**	
CNDGT	*dest, src1, src2*	if(*src1* > *src2*) else	*dest* = *src1*; *dest* = *src2*;
CNDLT	*dest, src1, src2*	if(*src1* < *src2*) else	*dest* = *src1*; *dest* = *src2*;
CNDEQ	*dest, src1, src2*	if(*src1* == *src2*) else	*dest* = *src1*; *dest* = *src2*;
CNDNE	*dest, src1, src2*	if(*src1* != *src2*) else	*dest* = *src1*; *dest* = *src2*;
CNDEZ	*dest, src1, src2*	if(*src1* == 0) else	*dest* = *src1*; *dest* = *src2*;
CNDNZ	*dest, src1, src2*	if(*src1* != 0) else	*dest* = *src1*; *dest* = *src2*;
MADD1	*dest, src1, src2*	*dest* = *src1* * *src2* + *src1*;	
MADD2	*dest, src1, src2*	*dest* = *src1* * *src1* + *src2*;	
MSUB1	*dest, src1, src2*	*dest* = *src1* * *src2* - *src1*;	
MSUB2	*dest, src1, src2*	*dest* = *src1* * *src1* - *src2*;	

Figure 7.1 shows new functional units which are added into a single processing element. For the combined conditional operations, a function unit, including the function-

ality of one comparator and one transfer module is defined. Another function unit for the combined multiplication operations includes one multiplier and one adder.



(a) Functional unit for conditional operation



(b) Functional unit for ALU operation

Figure 7.1 Functional units for new instruction sets

7.3 PERFORMANCE ANALYSIS OF NEW INSTRUCTION EXTENSIONS FOR CONGESTION CONTROL APPLICATIONS

For this experiment, instruction annotation technology is used based on Simplescalar PISA instruction sets and cross compiler (gcc-pisa), and hand-coded optimized versions of each congestion control applications. As shown in Table 7.2, 4 different hard-

ware configurations are used for a single processing element: 4-way, 4-way+extension, 8-way and 8way+extension. In order to support new instruction extensions, two CND functional units and two MADD / MSUB functional units are added into 4-way baseline architecture. The 8-way model doubles the issue width, the execution modules and memory bandwidth.

Table 7.2 Architectural configurations for performance analysis

Architectural element	4-way	4-way + ext	8-way	8-way + ext
Issue width	4	4	8	8
ALU	4	4	8	8
Memory Port	2	2	4	4
CND	0	2	0	4
MADD/MSUB	0	2	0	4

As shown in Figure 7.2, the performance improvement from instruction extensions provides 10~12% which is not a big improvement, so exploiting packet level parallelism is really required to make desired throughput as well. It is observed that the RED shows more sensitivity from increasing issue widths, and new extensions show more performance improvement from the wider issue width.

7.4 SUMMARY

In this chapter, application-specific acceleration techniques are proposed to exploit instructions for congestion control applications. From the ILP perspective, new instruction set extensions for fast conditional operations are applied for congestion control

applications. Based on experiments, proposed architectural extensions show 10~12% improvement in performance for instruction set enhancements.

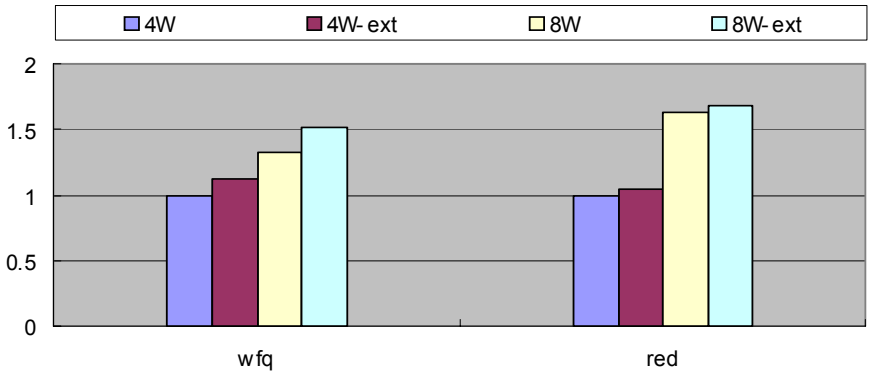


Figure 7.2 Performance evaluation of new instruction extensions

Chapter 8: Conclusions and Future Work

As modern network technologies have explosively grown along with Internet businesses, various applications and protocols require intelligent processing over the network. Therefore, network interfaces also have to keep up with the speed, throughput and capability to support all the workloads over the network. Good understanding of the target applications from the architectural perspectives is essential in designing a network processor.

The network processor on the physical port of a router should be able to process the modern workloads without slowing down line speed. Computational load per packet (in terms of number of instructions) ranges from 880 to 4,800, when assuming a stream of minimum-sized packets of 64 bytes and a single processor of 1 GHz clock frequency. The required parallelism for executing NP applications is in the range of 2 ~ 356 [64]. Hence a conventional processor is not enough to handle these workloads. Relevant research is required to identify appropriate architectures to efficiently execute these emerging workloads.

Control plane network applications that handle traffic management, quality of service etc. have become very important. While most of the previous research and commercial products for NPs are dedicated to data-plane applications, control-plane applications are not well understood. With the demands of these emerging network applications, it is imperative to develop and quantitatively characterize the NP control plane workloads to guide architects for designing future NPs.

In this dissertation, a new benchmark suite for network processors is proposed and its architectural workload characteristics are presented. Parallelism characteristics of these applications are analyzed and hardware acceleration techniques are proposed as alternative solution of existing architectures to deal with new emerging workloads.

8.1 CONCLUSIONS

This dissertation makes several contributions to the defining of a new network processor benchmark suite, characterization of network processor workloads, the detection of bottlenecks in network workloads, and towards designing architectural alternatives including instruction set extensions, and hardware acceleration. These are also described in more detail in [62][64][65][66]. The summary of the contributions is listed below:

1. A set of benchmarks, called NpBench targeted towards control plane workloads as well as data plane workloads, is proposed. While previously released network processor benchmarks deal with data plane applications, the NpBench includes emerging control plane applications. With the increasing demand of QoS and rapidly changing modern network environments, the significance of control plane workloads will be becoming larger. NpBench includes 5 control plane applications and 5 data plane applications. Control plane applications consist of Weighted Fair Queuing (WFQ), Random Early Detection (RED), Secure Sockets Layer Dispatcher (SSLD) and Media Transcoding (MTC). Data plane applications consist of Advanced Encryption Standard (AES), Message Digestion (MD5), Diffie-Hellman

(DH), Packet Fragmentation (FRAG) and Cyclic Redundancy Check (CRC). The NpBench suite is implemented using C and is opened to public [64]. Large number of institutions in the world have licensed and several papers and articles cite the NpBench [19][49][77][78][86] [88][111][116][118][119].

2. The characteristics of network processor workloads such as instruction mix, cache behavior, available parallelism and required processing capability per packet are presented and compared with existing benchmark suites. Architectural characteristics of the application having control plane properties are presented along with their implications to designing network processors and the significance of additional parallelism to perform NP applications at wire speed. From the experiments, it is seen that branch operations are heavily used for executing control plane applications. Control plane workloads are seen to have more ILP than data plane workloads. Based on the analysis of available parallelism, NP architectures still need more parallel implementations with instruction level parallelism (ILP) within the PEs or control plane processors. This contribution is described in more detail in [64].
3. Parallelism characteristics of network processing applications are compared to multimedia applications. NP applications (e.g. WFQ) have both parallelizable operations and a significant amount of serial operations, while most of multimedia applications have large amount of parallelizable operations. Network processor applications are quite different from multimedia and DSP applications in architectural aspects, but both applications have large parallelism and iteratively utilize many similar algorithms. Based on this investigation, it can be concluded that the

architectures successfully used in multimedia domain can be a good candidate for the architectures of NP applications.

4. It is investigated that whether the success of VLIW in the multimedia field can be applied to the network processor domain as a processing element for a parallel architectural implementation. This premise is analyzed through the comparison between network processor workloads and multimedia workloads in terms of performance (speedup) and power consumption. It is found that NP applications need more aggressive optimization techniques in static scheduled architecture, while media applications can get large parallelism with simple basicblock optimizations. With the characteristics of large packet-level parallelism, experimental analysis supports static scheduling as an applicable paradigm for network processor applications with lower hardware complexity and lower power dissipation. Based on this investigation, it can be observed that the parallel implementation costs can be reduced by 50%~75% when applying VLIW to network processing. This contribution is described in more detail in [59].
5. Congestion applications have both parallelizable operations and same serial operations. In order to get large parallelism in the congestion applications, a hardware acceleration technique is introduced by decoupling the dataflow into the front-end PLP (Packet level parallelism) module and the back-end hardware acceleration module. When applying 16 ~ 64 acceleration module in parallel, 10x ~ 50x performance improvement can be obtained. This contribution is described in more detail in [65].

6. The procedure of media transcoding consists of transcoding decision unit and pipeline-styled regular processing units. The decision module handles which module is enabled and how many modules are enabled. Data conversion modules include several different kinds of functionalities. Based on the characterization experiments, data conversion module takes 89% of the total execution cycles. Most of data to be processed in the transcoding proxy are related to multimedia applications, so each processing unit requires regular and fine-grained processing element. Therefore, the array-style processor would be a good candidate for accelerating media transcoding. Array-style acceleration technique is proposed for data conversion module of media transcoding applications. If 64 parallel implementation of the acceleration module is applied, 28.7x transcoding speedup can be obtained. This contribution is described in more detail in [60].
7. Both SSLD and MPLS have the lookup table searching and updating module in each procedure. Lookup table searching is a major bottleneck in the LUT-related applications (77% ~ 86% of the total execution cycles in MPLS and SSLD). In this dissertation, acceleration techniques are proposed using this partitioned lookup mechanism for searching LUT used in MPLS and SSLD. Parallel comparator (for equality checking) is used to find exactly matched session ID (for SSLD) or FEC (forwarding equivalence class) ID (for MPLS) among partitioned LUTs. If 16-way parallel implementation of the acceleration module is applied, 262x improvement for MPLS and 362x for SSLD can be obtained. This contribution is described in more detail in [60].

8. The performance improvement of a single processing element is an important factor in designing parallel architecture. New instruction-set extensions are introduced to support fast execution of network processor applications, based on the detailed kernel analysis. Frequently used instructions are combined into one instruction. For congestion control applications, two groups of instruction sets are defined: conditional operations and multiplication-add / multiplication-sub. Proposed architectural extensions show 10~12% improvement in performance for instruction set enhancements [65].

8.2 FUTURE WORK

Several kinds of workloads and benchmarks have been developed as target applications for application-specific processors. While SPEC is working on the benchmarks of general purpose processors, EEMBC and several academic benchmark suites deal with the workloads of application-specific processors. However, these benchmark suites consist of several single workloads. They could not reflect realistic workload, since many network and multimedia applications are composite-type. In order to design an appropriate processor for a specific application, it is required to consider these evolving trends.

The processor architecture for network nodes has changed from software implementations on general purpose processors to ASICs and network processors. As the performance of general purpose processor rapidly increases, one may need to consider going back to use general purpose processors for network nodes, but more application-specific

extension modules are required to commonly used for network processor applications. For future work, the feasibility of extracting common architectural extensions over variety of network workloads, and defining architectural extensions (e.g., as in MMX for multimedia applications) can be investigated. Careful analysis of the workload will be conducted to identify more appropriate candidates for hardware extensions. This research can be extended in the following ways in the future:

1. Finding Common Hardware Acceleration Solutions for Emerging Network Processor Applications: Current network processors handle specific network applications at individual application level. This is because there is no common solution for network processing workloads and many companies want to use their own design solutions to avoid any costs for licensing fee and to reduce the time-to-market. There is no research on finding common architectural characteristics on network processor workloads, while several architectural solutions are commonly used for multimedia workloads (e.g., MMX, 3DNow!, etc). However, network processing is stream processing and it performs same patterns of processing with irregular patterns of data. So, if one can find common factors among several network applications, it can reduce the cost of development. Also, when this mechanism can be applied to general purpose processors, a hardware extension for NP applications can be developed as in MMX for multimedia application.

2. Extracting smallest set of NP benchmarks through the statistical analysis: As network environments keep evolving, extracting new workloads would be sig-

nificant in the future. However, it is not realistic to characterize all applications, and the relevant simulation time will be additional big bottleneck in the design procedures of a processor. Therefore it would be valuable work to design a framework to statistically analyze new emerging NP workloads and extract smallest set of benchmarks, so that simulation time can be reduced and time to market (TTM) can be significantly reduced.

3. Development of an Automated Framework that supports Performance Modeling and Design: Designing millions (or billions) of gates in RTL takes too long and too hard to verify its design. So many companies are making some efforts on performance modeling, to reduce development periods, which is to evaluate the target processor by software design of instruction-set simulator. Currently, several chip companies are working on this area, but there is no standardized solution for processor description and software design methodology. The instruction simulators can be automatically generated by the processor descriptions and the automated methodology. Since processors are frequently upgraded based on the previous version, once a version of processor descriptions have been developed, it can be easily developed new version of instruction-set simulator with small changes of descriptions. This area will be important as the demands of SoC development increases.

Bibliography

[1] C. Adams, "Constructing Symmetric Ciphers Using the CAST Design Procedure," *Designs, Codes and Cryptography*, vol. 12, pp. 283-316, Nov., 1997

[2] A. Agarwal, B. Lim, D. Kranz and J. Kubiatowicz, "A Processor Architecture for Multiprocessing," *Proc. of 17th International Symposium on Computer Architecture*, pp. 278 –288, 1990

[3] Agere Inc., *The Challenge for Next Generation Network Processors*, Agere Inc. white paper, 1999

[4] V. Agarwal, M. S. Hrishikesh, S. W. Keckler and D. Burger, "Clock Rate versus IPS: The End of the Road for Conventional Microarchitectures," *Proc. of the 27th Annual International Symposium on Computer Architectures*, pp. 248 –259, 2000

[5] Agilent Technologies, *JTC 003 Mixed Packet Size Throughput*, The Journal of Internet Test Methodologies, ed. 2.2, http://advanced.comms.agilent.com/n2x/docs/journal/JTC_003.html, 2002

[6] M. J. Akhbarizadeh and M. Nourani, "An IP Packet Forwarding Technique Based on Partitioned Lookup Table," *Proc. of the IEEE International Conference on Communications* (ICC'02), pp. 2263-2267, 2002

[7] G. Apostolopoulos, D. Aubespin, V. Peris, P. Pradhan and D. Saha, Design, "Implementation and Performance of a Content-Based Switch," *Proc. of INFOCOM'00*, pp. 1117-1126, 2000

[8] G. Apostolopoulos, V. Peris, P. Pradhan and D. Saha, *L5: A Self Learning Layer-5 Switch*, IBM research Report, RC21461, Apr. 1999

[9] G. Armitage, "MPLS: The Magic Behind the Myths," *IEEE Communications Magazine*, vol. 38, pp. 124-131, Jan. 2000

[10] T. M. Austin and G. S. Sohi, *TETRA: Evaluation of Serial Program Performance on Fine-grain Parallel Processors*, University of Wisconsin technical report #1162, 1993

[11] J. Bennett and H. Zhang, "Worst Case Fair Weighted Fair Queuing," *Proc. of IEEE INFOCOM* 95, pp. 120-128, 1995

[12] R. Bhargava, L. K. John, B. L. Evans and R. Radhakrishnan, "Evaluating MMX Technology Using DSP and Multimedia Applications," *Proc. of the IEEE Symposium on Microarchitecture*, pp. 37-46, 1998

[13] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, *An Architecture for Differentiated Services, RFC2475*, Internet Engineering Task Force (IETF) Network Working Group, http://www.ietf.org 1998

[14] J. M. Blanquer and B. Ozden, "Fair Queuing for Aggregated Multiple Links," *Proc. of ACM SIGCOMM 2001*, pp. 189-197, 2001

[15] B. Branden, *et al.*, *Recommendations on Queue Management and Congestion Avoidance in the Internet*, IETF Internet Draft, 1997

[16] D. Brooks, V. Tiwari and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," *Proc. Of 27th International Symposium on Computer* Architecture (ISCA), pp. 83-94, 2000

[17] D. C. Burger and T. M. Austin, "The SimpleScalar Toolset," *Computer Architecture News*, vol. 25, pp. 13-25, 1997

[18] W. Bux, W. E. Denzel, T. Engbersen, A. Herkersdorf and R. P. Luijten, "Technologies and Building Blocks for Fast Packet Forwarding," *IEEE Communication Magazine*, vol. 39, pp. 70-77, Jan. 2001

[19] M. Castelino and F. Hady, *Tutorial on NPF's IPsec Forwarding Benchmark*, Network Processing Forum, http://www.commsdesign.com/showArticle.jhtml?articleID=49400850, 2004

[20] P. P. Chang, S. A. Mahlke, W. Y. Chen, N. J. Water and W. W. Hwu, "IMPACT: An Architectural Framework for Multiple-Instruction-Issue Processors," *Proc. of the 18th Annual International Symposium on Computer Architecture,* pp. 266-275, 1991

[21] Cisco Systems, *Cisco IOS software and Multiprotocol Label Switching*, http://www.cisco.com/, Apr. 2000

[22] Cisco Systems, *Parallel eXpress Forwarding in the Cisco 10000 Edge Service Router*, White Paper, 2000

[23] Cisco Systems, *Release 12.4T New Features and Hardware Support,* No. 3001, http://www.cisco.com/, 2005

[24] ClearSpeed, *ClearSpeed MTAP processor (CSX600),* http://www.clearspeed.com, 2004

[25] R. F. Cmelik and D. Keppel, *Shade: A Fast Instruction-set Simulator for Execution Profiling*, Technical Report SMLI TR-93-12, SUN Microsystems Inc., 1993

[26] D. Comer and D. Stevens, *Internetworking with TCP/IP,* vol. II, Upper Saddle River, NJ: Prentice Hall, 1994

[27] *C-Port Network Processors (C-5, C-3e, C-5e, M-5),* http://www.freescale.com/

[28] P. Crowley, M. E. Fiuczynski, J-L Baer and B. Bershad, "Workloads for Programmable Network Interfaces," *Workload Characterization for Computer System Design*, Norwell, MA: Kluwer Academic Publishers, pp. 135-147, 2000

[29] S. Deering, and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification, RFC2460*, Internet Engineering Task Force (IETF) Network Working Group, 1998

[30] A. Demers, S. Keshav and S. Shenker, "Analysis and Simulation of a Fair Queuing Algorithm," *Proc. of ACM SIGCOMM*, pp. 1-12, 1989

[31] N. Deshpande, *TCP Extensions for Wireless Networks*, Student Report, Washington University in St. Louis, available at ftp://ftp.netlab.ohio-state.edu/pub/jain/courses/cis788-99/tcp_wireless.pdf

[32] K. Diefendorff and P. K. Dubey, "How Multimedia Workloads will Change Processor Design," IEEE *Computer*, vol. 30, pp. 43-45, 1997

[33] N. Doraswamy and D. Harkins, *IPSec: the New Security Standard for the Internet, Intranets, and Virtual Private Networks*, Upper Saddle River, NJ: Prentice Hall, 1999

[34] T. Eklund, *The World's First 40Gbps (OC-768) Network Processor*, Network Processor Forum, 2001

[35] The Electronic Design, *Chip Sets Hurdle QoS Issues To Deliver Wireless Video*, Nov. 2003, http://www.elecdesign.com/Articles/Index.cfm?ArticleID=6825

[36] *Embedded Microprocessor Benchmarking Consortium*, http://www.eembc.org

[37] EZchip Technologies, *Network Processor Designs for Next-Generation Networking Equipment*, White paper, 1999

[38] W. Fang, *Building An Accounting Infrastructure for the Internet*, Princeton University Computer Science Technical Report, TR-599-99, 1999

[39] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE Transactions on Networking*, vol. 1, pp. 397-413, 1993

[40] C. Fraleigh, C. Diot, S. Moon, P. Owezarski, D. Papagiannaki and F. Tobagi, Experiences Monitoring Backbone IP Networks, *Proc. of Workshop on Passive and Active Measurements on the Internet*, Amsterdam, The Netherlands, pp. 12-22, 2001

[41] J. Fritts and W. Wolf, "Evaluation of Static and Dynamic Scheduling for Media Processors," *Proc. of 2nd Workshop on Media Processors and DSPs,* pp. 33-43, 2000

[42] J. Fritts, W. Wolf and B. Liu, "Understanding Multimedia Application Characteristics for Designing Programmable Media Processors," *Proc. of The International Society for Optical Engineering (SPIE)*, pp. 2-13, 1999

[43] P. Gupta and N. McKweown, "Algorithms for Packet Classification," *IEEE Network*, vol. 15, pp. 24-32, 2001

[44] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown, "MiBench: A Free, Commercially Representative Embedded Benchmark Suite," *Proc. of the 4th annual IEEE International Workshop on Workload Characterization*, pp. 3-14, 2001

[45] D. Hammerstrom and E. Davidson, "Information Content of CPU Memory Referencing Behavior," *Proc. of the 4th International Symposium on Computer Architecture*, pp. 184-192, 1977

[46] R. Han, *et al*., "Dynamic Adaptation in An Image Transcoding Proxy for Mobile Web Browsing," *IEEE Personal Communications Magazine*, vol. 5, pp. 8-17, 1998

[47] G. J. Hekstra, G. D. La Hei, P. Bingley and F. W. Sijstermans, "TriMedia CPU64 Design Space Exploration," *Proc. of International Conference on Computer Design*, pp. 599-606, 1999

[48] J. L. Hennessy and D. A. Patterson, *Computer Architecture - A Quantitative Approach*, 3rd ed., San Francisco, CA: Morgan Kaufmann Publishers, pp. 363-367, 1995

[49] A. Heppel, *An Introduction to Network Processors*, White paper, Roke Manor Research Ltd., 2003

[50] C-H. Hsu and U. Kremer, "The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction," *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'03)*, pp. 38-48, 2003

[51] IBM Corp., *IBM Network Processor (IBM32NPR161EPXCAC100)*, Product Overview, 1999

[52] *Intel IXP1200 Network Processor*, http://www.intel.com/design/network/products/npfamily/ixp1200.htm

[53] S. Keshav and R. Sharma, "Issues and Trends in Router Design," *IEEE Communications Magazines*, vol. 36, pp. 144-151, May 1998

[54] M. Kovac and N. Ranganathan, "JAGUAR: A Fully Pipelined VLSI Architecture for JPEG Image Compression Standard," *Proc. of the IEEE*, vol. 83, pp. 247-258 1995

[55] H. T. Kung, "Why Systolic Architectures?" *IEEE Computer*, vol. 15, pp. 37-46, 1982

[56] *LDP Specification,* Internet Engineering Task Force (IETF) Network Working Group, http://www.ietf.org/, 2001

[57] J. Law and B. K. Lee, *Access Time and Power Characteristics of Various Future File Configurations*, Technical Report TR-020821-01, Laboratory for Computer Architecture, The University of Texas at Austin, 2002

[58] B. K. Lee and L. John, *Development and Characterization of Control-plane Network Workloads*, Technical Report TR-030827-01, Laboratory for Computer Architecture, The University of Texas at Austin, 2003

[59] B. K. Lee and L. John, *Exploiting Statically Identified Parallelism for Network Processor Applications*, Technical Report LCA-TR-050615-02, Laboratory for Computer Architecture, The University of Texas at Austin, 2005

[60] B. K. Lee and L. John, *Hardware Acceleration Techniques for Control-plane Applications in Network Processors*, Technical Report LCA-TR-050615-01, Laboratory for Computer Architecture, The University of Texas at Austin, 2005

[61] B. K. Lee and L. K. John, "Implications of Executing Compression and Encryption Applications on General Purpose Processors," *IEEE Transactions on Computers*, vol. 54, pp. 917-922, 2005

[62] B. K. Lee and L. K. John, "Implications of Programmable General Purpose Processors for Compression / Encryption Applications," *Proc. of IEEE 13th International Conference on Application-specific Systems, Architectures and Processors*, pp. 233-242, 2002

[63] B. K. Lee and L. John, *Implications of Programmable General Purpose Processors for Compression/Encryption Applications*, Technical Report LCA-TR-020315, Laboratory for Computer Architecture, The University of Texas at Austin, 2002

[64] B. K. Lee and L. K. John, "NpBench: A Benchmark Suite for Control Plane and Data Plane Applications for Network Processors," *Proc. of the International Conference on Computer Design* (ICCD'03), pp. 226-233, 2003

[65] B. K. Lee, L. K. John and E. John, "Architectural Support for Accelerating Congestion Control Applications in Network Processors," *Proc. of IEEE 16th International Conference on Application-specific Systems, Architectures and Processors* (ASAP), pp. 169-175, 2005

[66] C. Lee, M. Potkonjak and W. H. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems," *Proc. of International Symposium on Microarchitecture*, pp. 330-335, 1997

[67] R. B. Lee, "Multimedia Extensions for General-purpose Processors," *Proc. IEEE Workshop on Signal Processing Systems*, pp. 9-23, 1997.

[68] W. Liao and L. He, "Power Modeling and Reduction of VLIW Processors," *Proc. of Workshop on Compilers and Operating Systems for Low Power*, pp. 155-171, 2001

[69] Linley Gwenap, "Net Processor Makers Race toward 10-Gbit/s Goal," *EE Times*, June 19, 2000, available at http://www.eetimes.com/story/OEG20000619S0011

[70] H. Liu, "A Trace Driven Study of Packet Level Parallelism," *Proc. of International Conference on Communications* (ICC), pp. 2191-2195, 2002

[71] Lucent Technologies, Inc., *PayloadPlus TM Fast Pattern Processor*, http://www.agere.com/support/non-nda/docs/FPP-ProductBrief.pdf, 2000

[72] J. Matthew, "Network Processor Companies Face the Same Tough Issues," *Electronic News Online*, available at http://www.electronicnews.com/enews/Issue/RegisteredIssues/2000/07172000/z24f-1.asp, July 17, 2000

[73] Media Transcoding and Optimization - LightSurf 6 Open Standards MMS Platform, LightSurf Technologies Inc., http://www.lightsurf.com/technology/optimization.html

[74] S. Melvin, M. Nemirovsky, E. Musoll, J. Huynh, R. Milito, H. Urdaneta and K. Saraf, "A Massively Multi-threaded Packet Processor," *Proc. of Workshop on Network Processors - NP2 held in conjunction with Ninth International Symposium on High Performance Computer Architecture (HPCA-9), Anaheim, CA,* pp. 64-74, 2003

[75] G. Memik and W. H. Mangione-Smith, "Increasing Power Efficiency of Multi-Core Network Processors Through Data Filtering," *Proc. of International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, pp. 108-116, 2002

[76] G. Memik, W. H. Mangione-smith and W. Hu, "NetBench: A Benchmarking Suite for Network Processors," *Proc. of International conference on Computer Aided Design (ICCAD),* pp. 39-42, 2001

[77] Y. L. Moullec, N. B. Amor, J. Diguet and P. Koch, "Adaptive Wireless Systems Optimization based on Follow-up Modeling," *GSPx 2004:Embedded Applications Software & Hardware*, 2004

[78] Y. L. Moullec, N. B. Amor, J. Diguet and P. Koch, "Follow-up Modeling for Wireless Personal Communication Systems," *Proc. of Wireless Personal Multimedia Communications (WPMC), Abano Terme, Italy,* pp. 255-259, 2004

[79] *Multi Protocol Label Switching Architecture,* Internet Engineering Task Force (IETF) Network Working Group, available at: http://www.ietf.org/, 2001

[80] A. Nemirovsky, *Towards Characterizing Network Processors: Needs and Challenges*, Xstream logic, white paper, 2000

[81] M. Nemirovsky, *Simultaneous Multithreading Architectures: Enabling the Next-Generation Internet*, XStream Logic Devices, 2000

[82] Network Processing Forum, *Network Processing Forum Unveils Specifications Roadmap*, Press Release, April 30, 2001

[83] H. Nguyen and L. K. John, "Exploiting SIMD Parallelism in DSP and Multimedia Algorithms using the AltiVec Technology," *Proc. of ACM International Conference on Supercomputing*, pp. 11-20, 1999

[84] K. Nichols, S. Blake, F. Baker and D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers,* Internet Engineering Task Force (IETF) Network Working Group, available at: http://www.ietf.org/, 1998

[85] *NpBench Website*, http://lca.ece.utexas.edu/npbench/

[86] *NpBench*, http://www.cc.gatech.edu/classes/AY2005/cs8803hpc_spring/

[87] P. Pappu and T. Wolf, "Scheduling Processing Resources in Programmable Routers," *Proc. of the Twenty-First IEEE Conference on Computer Communications* (INFOCOM), pp. 104 –112, 2002

[88] R. Ramaswamy, N. Weng, and T. Wolf, "Analysis of Network Processing Workloads," *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software* (ISPASS), pp. 226-235, 2005

[89] N. Ranganathan and S. Venugopal, "A VLSI Chip for Template Matching," *Proc. of IEEE International Symposium on VLSI Design*, pp. 542-545, 1994

[90] N. Ranganathan and S. Henriques, “High-Speed VLSI Designs for Lempel-Ziv-Based Data Compression,” *IEEE Transactions on Circuits and Systems*, pp. 96-106, 1992

[91] P. Ranganathan, S. Adve and N. Jouppi, “Performance of Image and Video Processing with General-purpose Processors and Media ISA Extensions,” *Proc. of 26th IEEE/ACM Symposium on Computer Architecture*, pp. 124-135, 1999.

[92] P. H. Rao and S. K. Nandy, “Evaluating Compiler Support for Complexity-Effective Network Processing,” *Proc. of Workshop on Complexity Effective Design* (WCED), pp. 39-42, 2003

[93] E. Rosen, A. Viswanathan and R. Callon, *Multiprotocol Label Switching Architecture,* Internet Engineering Task Force (IETF) Network Working Group, available at: http://www.ietf.org/, 2001

[94] E. Seamans and M. Rosenblum, “Parallel Decompositions of a Packet-Processing Workload,” *Proc. of Advanced Networking and Communications Hardware Workshop (ANCHOR) held in conjunction with the 31st Annual International Symposium on Computer Architecture (ISCA 2004), Munich, Germany,* pp. 40-48, 2004

[95] *Security Architecture for the Internet Protocol,* Internet Engineering Task Force (IETF) Network Working Group, available at: http://www.ietf.org/, 1998

[96] N. Shah, *Understanding Network Processors*, Master's thesis, University of California, Berkeley, 2001

[97] *SimpleScalar LLC*, available at: http://www.simplescalar.com

[98] G. A. Slavenburg, S. Rathnam, and H. Dijkstra, "The Trimedia TM-1 PCI VLIW Media Processor," *Proc. of Hot Chips VIII Symposium*, pp. 171-177, 1996

[99] N. T. Slingerland and A. J. Smith, "Cache Performance for Multimedia Applications," *Proc. of the 15th IEEE International Conference on Supercomputing*, pp. 204-217, 2001

[100] J. R. Smith, R. Mohan and C. Li, "Content-based Transcoding of Images in the Internet," *Proc. of IEEE Conference on Image Processing* (ICIP-98), pp. 7-11, 1998

[101] Y. Song and E. Aboelela, "A Parallel IP-Address Forwarding Approach Based on Partitioned Lookup Table Techniques," *Proc. of 29th Annual IEEE International Conference on Local Computer Networks* (LCN'04), pp. 425-426, 2004

[102] *SPEC CPU2000*, Standard Performance Evaluation Corporation (SPEC), http://www.spec.org/osg/cpu2000/

[103] *SSL Protocol* version 3.0, Internet Engineering Task Force (IETF) Transport Layer Security Working Group, http://wp.netscape.com/eng/ssl3/ssl-toc.html, 1996

[104] D. Stiliadis and A. Varma, "Efficient Fair-queuing Algorithms for Packet-switched Networks," *IEEE/ACM transactions on Networking*, pp. 175-185, 1998

[105] *Synopsys Design Vision - Products and Solutions,* http://www.synopsys.com/

[106] D. Talla and L. K. John, “Cost-effective Hardware Acceleration of Multimedia Applications,” *Proc. of the IEEE International Conference on Computer Design* '01, pp. 415-424, 2001

[107] D. Talla and L. K. John, “Execution Characteristics of Multimedia Applications on a Pentium II Processor,” *Proc. of the IEEE International Performance, Computing and Communications Conference*, pp. 516-524, 2000

[108] D. Talla and L. K. John, “Performance Evaluation and Benchmarking of Native Signal Processing,” *Euro-Par '99*, pp. 266-270, 1999

[109] D. Talla, L. K. John and D. Burger, “Bottlenecks in Multimedia Processing with SIMD Style Extensions and Architectural Enhancements,” *IEEE Transactions on Computers*, vol. 52, pp. 1015-1031, 2003

[110] D. Talla, L. K. John, V. Lapinskii and B. L. Evans, “Evaluating Signal Processing and Multimedia Applications on SIMD, VLIW and Superscalar Architectures,” *Proc. of the IEEE International Conference on Computer Design*, pp. 163-172, 2000

[111] Z. Tan, C. Lin, H. Yin and B. Li, “Optimization and Benchmark of Cryptographic Algorithms on Network Processors,” *IEEE Micro*, vol. 24, no. 5, pp. 55-69, Sep. 2004

[112] A. Tanenbaum, *Computer Networks*, Upper Saddle River, NJ: Prentice Hall, 1996

[113] *Ternary CAM*, SiberCore Technology, http://www.sibercore.com

[114] *TMS320C62X/C64X/C67X Reference Manual*, Texas Instruments, http://www.ti.com/cst

[115] *Trimaran toolset*, http://www.trimaran.org

[116] M. Tsai, C. Kulkarni, C. Sauer, N. Shah and K. Keutzer, "A Benchmarking Methodology for Network Processors," *Proc. of 1st Workshop on network processors*, pp. 63-74, 2002

[117] M. Valluri, L. John and H. Hanson, "Exploiting Compiler-generated Schedules for Energy Savings in High-performance Processors," *Proc. of the International Symposium on Low Power Electronics and Design* (ISLPED), pp. 414-419, 2003

[118] J. Verdú, J. García, M. Nemirovsky and M. Valero, *Analysis of Traffic Traces for Stateful Applications*, DAC Report-2003-53, Universitat Politècnica de Catalunya, 2003

[119] J. Verdu, J. García, M. Nemirovsky and M. Valero, "The Impact of Traffic Aggregation on the Memory Performance of Networking Applications," *Proc. of Workshop on Memory Performance, Dealing with Applications, Systems and Architectures* (MEDEA), pp. 59-64, 2004

[120] J. Williams, "Architectures for Network Processing," *Proc. of IEEE International Symposium on VLSI Technology, Systems, and Applications*, pp. 61-64, 2001

[121] S. J. E. Wilton and N. P. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model," *IEEE Journal of Solid-State Circuits*, vol. 31, pp 677-688, 1996

[122] T. Wolf and M. A. Franklin, "CommBench - A Telecommunications Benchmark for Network Processors," *International Symposium on Performance Analysis of Systems and Software*, pp. 154-162, 2000

[123] T. Wolf and M. A. Franklin, "Design Tradeoffs for Embedded Network Processors," *Proc. of International Conference on Architecture of Computing Systems* (ARCS) (Lecture Notes in Computer Science), pp 149 –164, 2002

[124] T. Wolf and M. A. Franklin, "Locality-aware Predictive Scheduling for Network Processors," *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software* (ISPASS), pp 152 –159, 2001

[125] T. Wolf and J. S. Turner, "Design Issues for High Performance Active Routers," *Proc. of the International Zurich Seminar on Broadband Communications*, pp 199 –205, 2000

[126] X. Xiao and L. M. Ni, "Internet QoS: A Big Picture," *IEEE Network*, vol. 13, Mar.-Apr., pp. 8-18, 1999

VITA

Byeong Kil Lee was born in Kyungpook, Korea, on March 11, 1967, as the first son of Jae Ho Lee and Cha Joo Bae. After completing his high school education at Neung In High School in Taegu, Korea, he entered the Department of Electrical Engineering, Kyungpook National University in Taegu, Korea in March 1985. He received the degree of Bachelor of Science in Electrical Engineering from Kyungpook National University in February 1989. He joined the graduate program for Electrical Engineering at same university in March 1989 and obtained the degree of Master of Science in Electrical Engineering in February 1991. From February 1991 to June 2000, he was a senior researcher at Agency for Defense Development in Korea. In September 2000, he entered the Ph.D. program in Electrical and Computer Engineering at The University of Texas at Austin. During July 2004 to May 2005, he interned at Texas Instruments Inc., Austin. He joined Texas Instruments Inc. as a full-time engineer from May 2005. He is a student member of IEEE, IEEE Computer Society and ACM.

Permanent address: Bosung APT 101-803 Upnedong Bukgu
Taegu, Korea 702-772

This dissertation was typed by the author.