ARCHITECTURE CHARACTERIZATION FOR PACKET PROCESSING

S.T.G.S.Ramakrishna, Abid Aquil, H.S.Jamadagni
CEDT, Indian Institute of Science
Bangalore, India.

ABSTRACT

Increasing network speeds have placed enormous burden on the processing requirements and the processors are expected to carry out a variety of tasks. Network Processors (NP)[1][2] is the blanket name given to the processors, which are evolved with a tradeoff, flexibility Vs performance. Network Processors are offered by a number of vendors; to take the main burden of processing requirement of network related operations from the conventional processors. Network Processors cover a spectrum of design tradeoff, that span in between the custom ASIC and the general-purpose processors. However the need is not yet well established and is undergoing continuous refinements. This paper focuses on the performance evaluation for network processors vis-à-vis general-purpose processors and determine if Network Processors have a role to play in design of network products in future.

Key Words: Network Processors, Performance Analysis, Packet processing, Multithreaded, IXP1200

1. INTRODUCTION

Complexity of communication systems has outpaced the technology evolution predicted by Moore's Law. The communication infrastructure demands and systems have stringent requirements and both have contradictory possibilities: (i) reducing cost in multi standard, evolutionary environment requires flexible and adaptive solutions. (ii) Performance requirement of systems demands for full custom solutions. This compels that the tradeoff is properly weighed to get good solutions.

Increasing use of the Internet [3] in different application domains, calls for fast access of data and this places stringent requirement on the nodes [4] in a network. Table 1 shows the available time for processing packet from ingress to egress for a minimum sized packet to meet the line rate access. The performance and flexibility requirement of nodes at different parts of the network is shown in table 2.

The paper is organized as follows, section 2 introduces the problem, section 3 describes briefly about the Algorithms chosen for benchmarking, section 4 presents the simulation results and section 5 presents the conclusions drawn from the results.

SONET rate	Bit Rate (Gbps)	No of Packets/sec (40 bytes)	Packet Duration (ns)
OC-48c	2.488	7,776,000	128.6
OC-192c	9.953	31,104,000	32.2
OC-768c	39.813	124,416,000	8

Table 1: Processing Requirements for different data transfer rates

Nodes	Performance	Flexibility
Core	High	Low
Edge	Medium	Medium
Access	Low	High

Table 2: Performance and Flexibility requirements for different network nodes.

2. PROBLEM FORMULATION

Processors designed for any domain have to meet certain market constraints apart from the processing requirements such as:

- Strict cost margin: Often, the design objective is not only to minimize cost but to reach a certain market segment and the needs they are addressing consisting of a price range and an expected functionality. So, if cost is above the intended price margin, functionality and performance may have to be redefined. This is different from application specific processors, which are designed with performance as the major requirement.
- Time-to-market and predictable design time: This constraint also applies to general-purpose processor design. Furthermore, the development can lead to a family of similar products or include other embedded products, which might not yet be completely defined.
- Flexibility: In a constantly evolutionary and adaptive environment this is the major factor, which differentiates it from the competing products and helps in reaching the market faster than others.

- Power dissipation: This is certainly a problem in the System on Chip (SoC) design because of usage of different design cores, which are optimized for different application.

Solutions[5] are proposed which are massively parallel and does the packet processing at wire speeds. Though a few NPs are in market, they have not assumed a central role in network processing. Moreover problems which are common in parallel processor environment like the compiler design, operating system, scheduling, interconnects and memory requirements have to be resolved while designing and using NPs. While new NPs are being designed, processing power of General-Purpose Processors (GPP) is also increasing many folds. In Dec 2002 Intel announced Pentium 4 processor, operating at 3.06 GHz. Use of single fast processor eliminates most of the problems and constraints described above. With the memory peak bandwidth [6] increasing continuously and the processing power of general-purpose processors increasing many folds[7], the question is (i) whether there is a need for network processors or will the existing processors sufficient to process the complexities involved in packet processing? Therefore a performance comparison of the evolving high speed GPPs and NP is essential to answer the above question and to establish the need for NP, if it is found superior for network operations.

3. ALGORITHMS FOR BENCHMARKING

In recent years, performance evaluation of processor and memory architectures [8] has become important because timely evaluation can impact procurement decisions, speedy deployment, and influence the design of future systems.

The standard way of performance evaluation of any system is by analysis or simulation. Analytical methods are good to get bounds on the resource required for operations and the number of computing elements such as processors. Moreover processors are difficult to model while incorporating all the parameters without simplifying assumptions. Therefore traditional way of evaluating performance of processors is by simulations. The standard way has been to execute a set of algorithms, specific to an application domain for which they are being evaluated. Simulation based approaches are good for small designs but as the design space increases it is slow and time consuming.

While a large number of algorithms can be chosen for performance evaluation of an NP [9], eight algorithms, which are part of packet processing, are selected in this paper. The selection of the algorithms is based on their complexity. They belong to various layers of TCP/IP stack. Choosing a right mixture of algorithms is important to get reliable performance metrics. Choice and description of the algorithms is given below:

FRAG: This algorithm fragments or splits a packet of size ≥1500 bytes based on the Maximum Transfer Unit (MTU) fixed for the network. This includes recalculation of the checksum, which forms a dominating part of fragmentation. Apart from this, the header information (i.e. source address, destination address) of the big packet is copied into smaller packets and the fields in the IP header are set appropriately so that the receiving terminal can reassemble the packets with out errors.

CRC: This algorithm calculates checksum based on cyclic redundancy check described in ISO 3309[10][11]. CRC-32 is widely used in Ethernet and ATM Adaptation Layer 5 checksum calculations. In this application a lookup table is created for all the possible combinations of a byte. The checksum of the packets are calculated taking a byte at a time, referring to the table for lookup and finally adding the checksum of all the bytes.

RTR: (Radix Tree Routing table lookup). Routing table lookups are important in packet processing where it is performed on each packet in datagram based network, and on each connection, in connection based network. This is available from the public domain NetBSD distribution [12]. RTR lookup is performed based on a data structure, which is built in the memory. This structure contains information regarding the next hop, MAC address of out going link, mask for prefix match. Best match is found for a given destination address based on the prefix search and that particular route is selected and the information is finally written into the packet header.

DRR: Deficit Round Robin scheduling [13] is a scheduling method implemented in several of the switches today for QoS (Quality of Service) guarantees. In this, each connection has a separate queue and the server tries to guarantee equal data being sent from each queue. This implementation is available from NetBench [14].

Deep Packet Searching (URL): An example of a deep packet searching is implementation of the URL-based switching, which is a commonly used context-switching mechanism. In this all the packets coming to a switch are parsed and switched according to the information content in the payload. Server load balancing is achieved in this application and subsequently it increases the utility of specialized servers in a server farm. The implementation is available from NetBench [14].

REED: This algorithm is a Forward Error Correcting code called Reed-Solomon scheme. This application mainly adds the redundancy to the existing data, so that even if the network corrupts the data due to transmission errors, it can be recovered from the redundant information present [15][16].

MD5: This is one of the network security algorithms based on one-way hash function that takes entire packet and from it computes the fixed length bit string. Message Digest Algorithm [17] creates a cryptographically secure signature for each outgoing packet. At the destination it is checked for the signature and if it does not matches, it implies the packet is hacked or corrupted [18] and accordingly the packet is dropped. The implementation is available from NetBench [14].

DH: Diffie-Hellman is common public key encryption-decryption mechanism. It helps in establishing a shared secret key[18]. It is the security protocol employed in most of the VPN (Virtual Private Networks). This implementation is available from NetBench [14].

3.1. Grouping of Algorithms

The algorithms described above can be grouped into two domains. FRAG, CRC, RTR and DRR algorithms are used in layers 1-3 where the operations are performed at bit/byte level. REED, MD5, DH, and Deep Packet Parsing(URL) algorithms are used in layers 4 and above of the TCP/IP stack and basic operations are performed at packet level.

Application layer algorithms such as ZIP, Virus scanning are considered as nonreal time operations. But network demands at the edges requires these operations to be performed in real time. Presently network processors are not targeted for those applications. JPEG, Speech Codecs are media based algorithms and there are processors called media processors designed specifically for these applications.

Benchmarks like NetBench [14], Commbench [16] are available and none of them are considered for evaluation of network processors by EEMBC [19], npforum [20], a standard body that defines the software, hardware interfaces and benchmarking because [21]:

(i) These benchmarks do not exploit all the architecture features available in NPs and therefore, they cannot characterize NPU effectively.

(ii) These algorithms do not range over all the layers of network operations.

We selected the above-mentioned algorithms, which are across different layers of TCP/IP stack and which overcome some of the limitations. These set of algorithms are not *unique* and a number of different combinations are possible.

4. SIMULATION RESULTS

Algorithms described above are executed on Intel's Pentium 4 (operating at 1.5 GHz with 400MHz of data bus, 8KB of L1D cache), RISC (MIPS R3000 operating at 200 MHz, with instruction issue width 4 and out of order execution) and VLIW (TI TMS320c64x operating at 1.1GHz). VTune 6.1, Simplescalar 3.0 and Code Composure Studio (CCS) 2.2 simulators are used to run the algorithms on the above mentioned processors respectively.

All the algorithms were run on the respective processors for ten thousand packets and the number of instructions executed and the execution time were recorded. Number of packets processed per second are calculated basing on the execution time and the processor speed. Packets with different sizes ranging from 64 bytes to 1500 bytes with an average packet size of 724 bytes are generated for this analysis. Algorithms are written in C language on Window platform for Pentium 4 and VLIW processors. Algorithms for RISC are written in C language on Linux platform. Table 3 gives the number of packets processed per second by different processors. The compiler effect and OS overhead cannot be ignored and as such no optimization is done to the code for evaluation on different processors.

Pkts/sec is taken as performance metric for comparison among different processors, because the test were performed offline and we need a metric which can characterize the processor details, like which processor resources is useful for achieving this throughput.

Algorithms	Pentium 4[1]	RISC	VLIW (c64x)[2]
CRC	63,808	22,812	8,871
FRAG	277,224	108,106	60,260
RTR	42,319	8,496	5,113
MD5	10,739	9,607	72
DH	6,073	2,927	**[3]
URL	20,204	370	**
DRR	49,827	13,193	19,054
REED	543	189	493

Table 3: Throughput (Pkts/sec) by different processors with avg. packet size of 724 bytes.

Figure 1 shows the Clocks Per Instruction (CPI) comparison of Pentium 4, RISC and VLIW. The algorithms were executed for ten thousand packets and the corresponding number of instructions executed and the execution time in terms of cycles are recorded. CPI is calculated by dividing execution cycles with the number of instructions. CPI for Pentium 4 is more than 1.5 and this is due to the micro code execution. The CPI for algorithms REED, URL, DH, MD5 is greater than 2.2 and this is due to the large percentage of Load/Store,

[1] Optimization can increase throughput.

[2] Throughput can be increased significantly by optimizing the code.

[3] Data is not available due to problems in CCS.

Branch and ADD instructions. The CPI for DRR is very high and this is due to execution of kernel code and the same is seen with respect to VLIW processor[4]. RISC is performing better due to: out of order execution and large instruction issue width and the achieved parallelism is 1.5 in these algorithms.

Pentium 4 performed better in lookup algorithms, RTR and URL due to the presence of trace cache, which stores the decoded instructions and thus eliminates the fetch and decoded stage. Due to the absence of integer multiplication unit, latency is caused due to execution of "IMUL" instruction on floating point multiplication unit. "SHL" is used in most of the layer 1 - 3 algorithms for bit/byte selection and this causes latency. Lot of pipeline stalls "PPro Mem Stall" occurred for MD5 and DH algorithms. This is due to reading from the memory location to which it is writing.

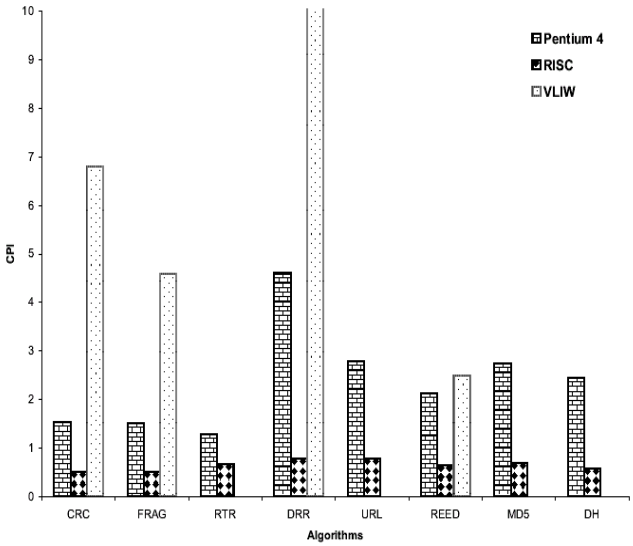

Figure 1: CPI comparisons

RISC performed better marginally in some of the algorithms due to out of order execution and small pipeline depth. Large instruction issue width helped for the parallel execution of the algorithms because these have a parallelism ≥1.5

The VLIW architecture is simple, where the scheduler packs a set of 8 instructions depending on the correlation between them and executes as one packet. So the focus in this type of architecture is more on the software part, and that differentiates them from other architectures. They are flexible and scalable. The analysis on DSP shows that it is not optimized for most of the operations and since it is targeted to multimedia and Forward Error Correcting (FEC) applications, there is significant improvement for the REED algorithm. We are unable to simulate algorithms URL and DH. Optimizing the code can enhance throughput significantly.

How far, processor's frequency of operation improves the individual processor performance is not analyzed because **qualitative** analysis is performed rather than the **quantitative** analysis. Instruction level comparison has to be made to comment on the individual performance.

4.1. Discussions

The packet-processing environment demands lots of parallelism in the architecture. The table 3 shows that the individual processors perform better for some of the algorithms based on Pkt/sec as a metric, for comparison. This analysis was done without considering the line rate and these results seems to defy that PIV performs better for most of the algorithms. To confirm the performance of these processors, scalability analysis is needed for higher line rates.

Table 4 and 5 represents the worst-case instruction complexity on Pentium 4 in terms of instructions/byte and instructions/packet respectively. Table 6 calculates the MIPS[5] required for 1 Gbps line rate and 40 Gbps line rate from the instructions per byte and instructions per packet numbers presented in tables 4 & 5 based on the equations (1) and (2).

Algorithms	Instructions/byte
CRC	1.76
FRAG	4.36
RTR	1.94
DRR	0.86

Table 4: Worst Case Instruction complexity for Layer 1-3 algorithms on Pentium 4 in terms of Instr/byte

Algorithms	Instructions/packet
MD5	1279
DH	2516
URL	3472
REED (enc & dec)	4257

Table 5: Worst Case Instruction complexity for Layer 4 and above algorithms on Pentium 4 in terms of Instr/pkt

[4] CPI for RTR and REED is not shown because they are more than 20

[5] MIPS for Pentium 4 is taken as 2881 from SiSoft Sandra CPU Benchmarks [22]

Algorithms	MIPS @ 1Gbps	MIPS @ 40 Gbps	No of Pentium 4 required @ 40Gbps
CRC	220	8,800	3
FRAG	545	21,800	8
RTR	243	9,700	4
DRR	108	4,300	2
MD5	221	8,833	3
DH	435	17,376	6
URL	600	23,978	9
REED (enc & dec)	735	29,399	10

Table 6: Scalability analysis of Pentium 4 @ 40 Gbps.

$$MIPS\_given\_linerate = \frac{Instr/byte \times linerate(bit/\sec)}{8(bit/byte)} \quad \textbf{(1)}$$

$$No\_of\_Pentium4 = \frac{MIPS\_given\_linerate(MIPS)}{Pentium4\_mips(MIPS)} \quad \textbf{(2)}$$

5. CONCLUSIONS

The MIPS column in table 6, shows that these algorithms individually can easily be executed on Pentium 4 at 1Gbps wire speed. The total MIPS requirement when all these algorithms need to be executed comes to around 2816 and this may be possible in Pentium 4 because no optimization is done. MIPS shown here does not take into account the control part i.e. protocol management of the packet processing. The MIPS requirement at higher line rates (40Gbps) requires more number of Pentiums and this is not feasible from system design point of view. The results presented here do not take into account the synchronization and communication MIPS.

Simulations and the scalability analysis showed that there is a requirement for special processors, as conventional processors cannot scale with increasing line rates. These results helped in establishing the following facts about general-purpose processors:

- I/O network data need to be tightly coupled with CPU execution so that, no time is wasted in ingress data movement, processing and egress data movement. Packet Processing involves intensive I/O operations and these processors are not designed to handle these events.
- While processing, a processor needs to keep track of several high-bandwidth input, output and memory access events concurrently. Therefore relaying on interrupts and single processor execution does not work, however fast a processor can operate. This is because the processors do not have the capability to hide these long latencies and Switch between various concurrent tasks rapidly.
- Caching of packets in a single processor is ineffective due to lack of locality and processor cycles are wasted in data movement.

Finally, performance comparison is made between IXP1200 (Intel's network processor)[6], Pentium 4 and RISC for two of the algorithms mentioned above, CRC and lookup. Figure 2 shows the throughput with one micro-engine of IXP1200.

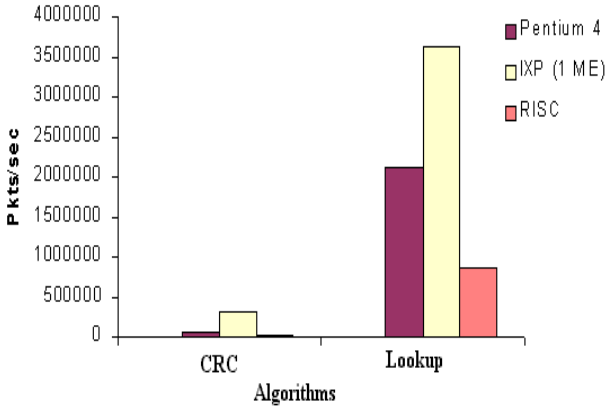

Figure 2: Performance comparison with IXP1200

6. REFERENCES

[1] Linda Geppert, "The New Chips on the Blocks", in IEEE Spectrum, January 2001.

[2] Werner Bux, Wolfgang E. Denzel, Ton Engbersen, Andreas Herkersdorf and Ronald P. Luijten, IBM Research "Technologies and Building Blocks for Fast Packet Forwarding", in IEEE Communications Magazine}, pp.70--74, January 2001.

[3] http://www.isc.org, July 2002

[4] Paul Rutten, Mickey Tauman, Hagai Bar-Lev and Avner Sonnino, "Is Moore's Law Infinite? The Economics of Moore's Law", *Kellogg Tech Venture* 2001 Anthology.

[5] David Husak, "Network Processors: A Definition and Comparison", White Paper, C-Port, A Motorola Company

[6] http://www.techweb.com

[7] http://public.itrs.net

[8] John L Hennessy and David A Patterson, "Computer Architecture, A Quantitative Approach".
Morgan Kaufmann Publishers Inc., 2nd Edition, 1996.

[6] Details are out of scope and cannot be presented in this paper.

[9] S.T.G.S.Ramakrishna, “Performance Evaluation for Network Processors”, MSc dissertation, Indian Institute of Science.
[10] International Organization for Standardization. ISO Information Processing Systems-Data Communication High-Level Data Link control Procedure - Frame Structure, 3rd Edition, IS 3309, October 1984
[11] “CRC-32 Calculation. Test Cases and HEC Tutorial”.
http://cell-relay.indiana.edu/cell-relay/publications/software/CRC/32bitCRC.c.html
[12] NetBSD Foundation Inc. [1998]. NetBSD 1.3 Release, January 4, 1998.
[13] M.Shreedhar and G.Varghese, “Efficient Fair Queuing using Deficit Round Robin”, in Proceedings of SIGCOMM'95}, Cambridge, MA, August-September 1995.
[14] Gokhan Memik, William H.Mangione-Smith and Wendong Hu, “NetBench: A Benchmarking Suite for Network Processors”, in Proceedings of International Conference on Computer Aided Design (ICCAD - 2001), pp.~39-43, November 4-8, San Jose, CA, 2001.
[15] T.R.N.Rao and E.Fujiwara, “Error-Control Coding for Computer Systems”. Prentice Hall, Englewood Cliffs, NJ, 1989.
[16] Tilman Wolf and M.Franklin, “CommBench - A Telecommunication Benchmark for Network Processors”, in Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp.~154-162, Austin, TX, April 2000.
[17] R.Rivest , “The MD5 Message-Digest Algorithm”, Request for Comment (RFC) 1321, April 1992.
[18] Andrew S. Tanenbaum, “Computer Networks”.
Prentice Hall of India, 3rd Edition, 1999.
[19] http://www.eembc.org
[20] http://www.npforum.org
[21] Prashant R.Chandra, Frank Hady, Raj Yavatkar, Tony Bock, Mason Cabot and Philp Mathew, “Benchmarking Network Processors”, in 8th International Symposium On High Performance Computer Architecture Workshop on Network Processors, Cambridge, Massachusetts, February 3, 2002.
[22] http://zdzich.wsh-kielce.edu.pl/~zbylu/teksty/piv_piii.html