Analytical Bounds on the Threads in IXP1200 Network Processor

S.T.G.S.Ramakrishna
Telematics Group, CEDT
Indian Institute of Science,
sramki@cedt.iisc.ernet.in

H.S.Jamadagni,
Chairman, CEDT
Indian Institute of Science,
hsjam@cedt.iisc.ernet.in

Abstract

Increasing link 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 traded for flexibility and 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. The Network Processors cover a spectrum of design tradeoff, that span in between the custom ASIC and the general-purpose processors. IXP1200 (Intel's network processor) is one among them. This paper focuses on deriving the analytical bounds on the optimum number of threads in IXP1200 at 1Gbps wire speed.

1. Introduction

Complexity of communication systems outpaces 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) Considering performance, full custom solutions are favored. This compels that the tradeoff should be properly weighed to get good solutions.

With increase use of Internet [3] in different application domains, fast access of data has become important and this places stringent requirement on the nodes [4] in a network. Table 1 shows the available time for processing packet from ingress to the egress for a minimum sized packet to meet the line rate access. Also nodes at different parts of the network have to meet performance and flexibility requirements as shown in table 2. Apart from this, there are a lot of non-functional market constraints such as Time to Market, Strict cost margin and flexibility has to be fulfilled by the design.

Table 1. Processing requirements for different data transfer rates

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 2. Performance and Flexibility requirements of network nodes

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

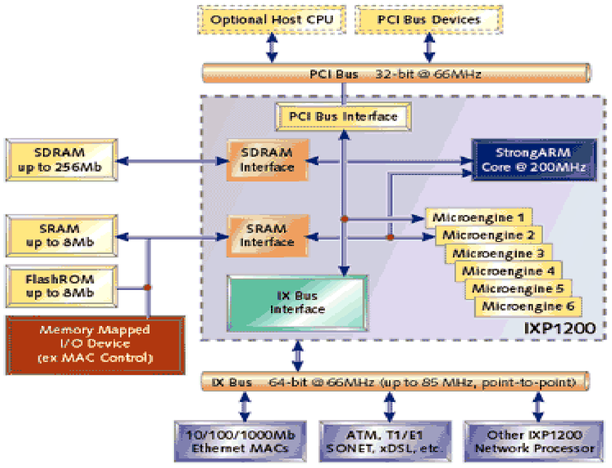
General Purpose Processors (GPP) cannot be scaled because networking operations requires:

- Fast, efficient and tight coupling of I/O events with CPU execution.
- There are potentially several high-bandwidth input, output and memory access events that a processor must keep track of concurrently. Relying on interrupts or a single thread of computation alone will not work no matter how fast a processor can operate because the processors do not have the capability to quickly multiplex between various concurrent tasks [5].

Multithreaded architectures [6] are favored for this domain because they explore locality (packet level, instruction level, algorithm level) available, to achieve the wire speed performance.

IXP1200 [7] is Intel's first generation network processor meant for 1Gbps wire speed operations. Figure 1 gives high-level details of the processor. It is an integrated processor with a strong ARM processor SA-1, six Microengines, standard memory interfaces and high-speed bus interfaces. All processors are fully and separately programmable and this makes IXP flexible for different network applications. This processor uses the locality principle (packets can be processed independently, without its history) in the

Internet and processes the packets in parallel with all the six microengines. The memory architecture is organized in hierarchy, of varying latencies, bandwidth and size. SDRAM (256MB) is meant for packet data storage, SRAM (8MB) for storing the lookup tables and important parameters, Flash ROM (8MB), Scratchpad (4KB) for storing the local variables, global variables and for storing the status words. It supports hardware multithreading with almost zero contexts swapping for parallel packet processing. This architecture hides the memory latency by overlapping the computing cycles with data access cycles. The large bus width helps in moving large amounts of data.



(Source: IXP1200 Documentation)

Figure 1: IXP1200 system block diagram

The paper is organized as follows, section 2 defines the problem, section 3 presents the assumptions and definitions of the variables used for the analysis, section 4 presents the analysis for the number of threads, and section 5 draws the conclusions basing on the work presented in this paper.

2. Problem formulation

For scalable architectures, communication latency (inter-processor, synchronization issues, I/O and data access requests) tends to increase with number of computing resources. So, the main challenge is to minimize the processor cycles wasted in long latency requests. Multithreading helps in hiding the latency. However in multithreading efficiency does not scale linearly with the number of thread.

IXP1200 uses four threads per micro-engine to make up a total of 24 threads per processor. So the question is (i) Why only four threads per micro-engine? (ii) Do four represent the optimum number of threads? (iii) What is the trend likely to be there for future network processors? These questions can be answered through simulations by executing the benchmark programs with varying number of threads. But the simulations results are very much dependant on the programs chosen for benchmarking and moreover as the workload increases, the system becomes slow and time consuming. Analytical methods are successfully applied to get good bounds on the resource required for operations, but a lot of simplifying assumptions about the design has to be made to make it worthwhile. In this paper we present, analytical method for deriving the bounds on the optimum number of threads in IXP1200 at 1Gbps line rate and project the same for future line rates.

3. Analytical solution

We analyze the IXP1200 to determine the optimum number of threads per micro-engine. IXP1200 is targeted at 1 Gbps line rate and is capable of processing 2.5 Million Packets per sec, assuming a typical packet size. The packet processing involves accesses to internal and external memories such as scratch memory, SRAM, DRAM, data movement from input and output FIFO buffers, etc. Number of memory accesses depends on the data access movement. The access latency (which depends on the number of requests) of SRAM is 30-50 cycles and DRAM is 100-150 cycles [8] at 232 MHz. These latencies represent the loaded conditions (i.e., when all the threads are requesting for data and the command queue is filled with requests). The hardware controlled multithreaded feature helps in hiding this latency.

Equations 1 and 2 compute the available micro-engine cycles for processing different packet sizes at different line rates.

$$Pkt\_intertime(ns) = \frac{PktSize(bytes) \times 8(bit/byte)}{line\_rate(bit/\sec)} \quad (1)$$

$$Cycles\_avail = \frac{Pkt\_intertime(ns)}{Processor\_cycletime(ns)} \quad (2)$$

As the line rate increases, the available computing cycles decreases for minimum size packet and increases for larger size packets. Increasing the speed of the processor can increase the number of cycles available per packet.

Memory access latency can also be expressed as number of arrival times of the minimum sized packet. The number of cycles to process a packet of 64 bytes at 1Gbps line rate is 103 cycles. While processing a packet, a processor requires access to both DRAM and SRAM. Therefore a processor with 103 cycles for processing a packet and all the threads doing memory access can do only one memory access with the latencies described above for DRAM and SRAM and is expressed by the equation:

$$\alpha = \lfloor T_{access} / N_A \rfloor \qquad (3)$$

Where $\mathbf{T_{access}}$ represents the latency for accessing the memory (DRAM & SRAM). α is the number of accesses to memory in a packet processing time. α takes only positive integer values, because a fractional values does not have any meaning for memory accesses and $\mathbf{N_A}$ represents the available processor cycles for processing a packet at a given line rate.

3.1. Assumptions

Before carrying the actual analysis, a number of ideal assumptions are made to derive the results and they are as follows:

- For modeling only one processor is assumed and issues involved with multiprocessors are ignored.
- Latency is due to machine structure and not due to the program behavior. Program behavior represents the access patterns of the instructions and the latency is assumed due to accessing the external memory.
- Outstanding requests are issued and it is assumed always there is a request in the pipeline, which needs to be served. This means that there is always a thread, which is ready and that can start execution in case idle period is encountered.
- Uniform distribution of threads is assumed and all threads are of equal priority

3.2. Definitions

A number of variables are used for the analysis and their definitions are as follows:

1. η: Number of threads.
2. **S**: Cycles lost in performing thread switching.
3. **B**: Busy period i.e., the interval between switching triggered by remote references.
4. $\mathbf{T_{access}}$: Latency while accessing external memory or by remote references.
5. ε: Efficiency.
6. $\mathbf{N_A}$: The available processor cycles for processing a packet at a given line rate.

In the following analysis, thread and context are used interchangeably.

4. Analysis

A single threaded processor waits during I/O accesses and this is said to be the idle period because the processor is not doing any computation. On the other hand, a multithreaded processor will suspend the current context which is doing the data access (latency operation) and switches to another context which is ready in the pipeline after a switching delay. Therefore, a processor with multiple contexts will hide the idle period (latency) by switching to a new context. Only if all the contexts are blocked, the processor suspends threads and will be in the idle state. Objective for high efficiency is to maximize the period for which it is busy. Basic efficiency equation for a processor is

$$\varepsilon = \frac{B}{B+I} \qquad (4)$$

In multithreaded processors the idle time is shared by switching time between threads and idle time due to multithreading.

Multithreaded processors exhibit three operating regions, (i) linear (efficiency is proportional to the number of threads), (ii) transition and (iii) saturation (efficiency depends only on the switching time between threads). Efficiency in linear region is less than saturation because there is no ready context available for execution and so the processor experiences idle cycles. This is given by equation 6:

$$\varepsilon_{linear} = \frac{\eta \times B}{B+S+I_\eta} \qquad (5)$$

The above equation shows, efficiency is directly related to η, but the efficiency cannot scale linearly with the number of threads because, as the level of sharing increases there are more threads to keep track of, at each decision point and more time is wasted per thread to administer the resource sharing among them. At this point the switching delay dominates and the efficiency decreases thereafter.

$$\varepsilon_{sat} = \frac{B}{B+S} \qquad (6)$$

Equation 6 represents the efficiency in saturation region. In saturation idle time is zero because there is always a context available for switching. The condition at which, transition [9] occurs from linear region to saturation region is, $\boldsymbol{\varepsilon_{linear} \geq \varepsilon_{sat}}$. Therefore solving this condition gives the following equation 8:

$$(\eta_{\min} - 1) \times (B+S) \geq I_\eta \qquad (7)$$

Where η_{min} represents the minimum number of threads for which this transition occurs. B (busy period) represents the switching interval between remote references and this can be approximated by N_A/η. S (switching time) is represented as, number of switchings multiplied by the latency per switching.

$$\gamma \times Latency\_per\_switching \qquad (8)$$

Where γ represents the number of switchings and this is related to number of threads as **η-1** [11]. Latency_per_switching represents the latency between context switching. Latency_per_switching for IXP1200 is 2-4 cycles [10]. I_η in Equation 7 represent the idle period with η threads, and this is due to latency. Substituting Equation 3 for I_η in Equation 8 and taking **α = 1** for the latencies of SRAM and DRAM at 1Gbps line rate. Substituting these in Equation 7 gives the following equation

$$(\eta_{\min} - 1) \times \left(\frac{N_A}{\eta_{\min}} + (\gamma \times 2)\right) - \alpha \times N_A \geq 0 \qquad (9)$$

γ is replaced by $\eta_{\min}$ - 1. The analysis is targeted for 1Gbps line rate; therefore N_A is 103 and taking α = 1 for the latencies (SRAM & DRAM) presented in section 3. Substituting these in Equation 10 gives

$$(\eta_{\min} - 1) \times \left(\frac{103}{\eta_{\min}} + ((\eta_{\min} - 1) \times 2)\right) - 103 \geq 0 \qquad (10)$$

This equation has to be satisfied for all possible combinations of latencies. Solving this equation taking Latency_per_switching as two gives $\eta_{\min}$ as **4.41**. Taking Latency_per_switching as four in Equation 8 and substituting these in equation 10 gives $\eta_{\min}$ as **3.66**. This analysis suggests **four** is the optimum number of threads per micro-engine in IXP1200 for a 64-byte packet at 1Gbps line rate. Figure 2 shows the thread requirements of IXP1200 at higher line rates. Number of threads required at respective line rates is calculated assuming 200MHz processor operation based on Equation 10.

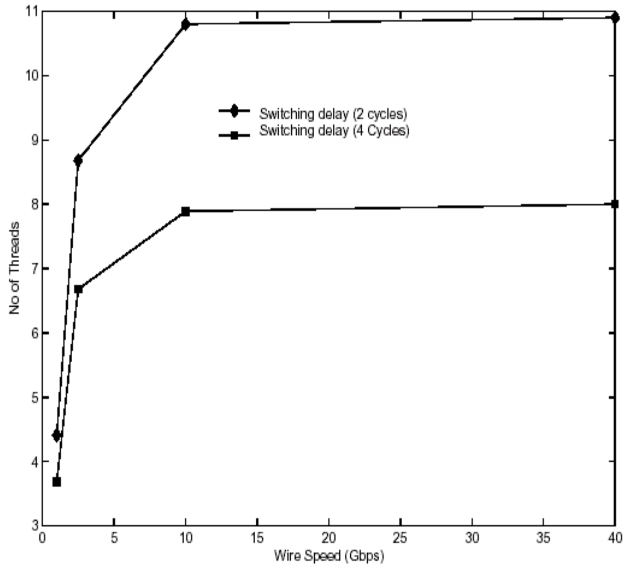


Figure 2: Threads calculations for different line rates.

5. Conclusions

Number of threads for a given line rate are always based on a typical packet size because the inter- packet arrival time are small compared to larger packet size. Therefore this analysis gives reliable number of threads that can be processed in a packet arrival time.

Analytical derivation shows four as the optimum number of threads at 1Gbps line rate. The number of threads remains constant after 10Gbps line rate and this is due to the available cycle for processing a packet equals the context switching delay. Therefore more number of threads at higher line rate is possible by increasing the available cycles, which is possible only by increasing the processor's frequency of operation. Therefore, future generation network processors operate at higher speed and have more number of threads per micro-engine.

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] S.T.G.S.Ramakrishna, "An Investigation on Architectural Elements of Network Processors", MSc, Thesis 2003, Indian Institute of Science.

[6] Patrick Crowley, Marc E. Fiuczynski, Jean-Loup Baer and Brain N. Bershad, "Characterizing Processor Architectures for Programmable Network Interfaces", in 2000 International Conference on Supercomputing, May 2000.

[7] IXP1200 Hardware Reference Manual.

[8] 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.

[9] Rafael H. Saavedra-Barrera, David E. Culler and Thorsten Von Eicken, "Analysis of Multithreaded Architectures for Parallel Computing".

[10] Erik Johnson and Aaron Kunze, "IXP1200 Programming, The Microengine Coding Guide for the Intel IXP1200 Network Processor Family", INTEL Press, 2002.

[11] James Bradley, "Under and Over Sharing Equations Relating Throughput Capacity, Resource-Sharing Level and the Source of Thrashing in Multi-threaded Systems", Research Report No: 2001-694-17, Department of Computer Science, University of Calgary, Canada.