Design Tradeoffs for Embedded Network Processors

Tilman Wolf[1] and Mark A. Franklin[2]

[1] Department of Computer Science,
Washington University, St. Louis, MO, USA
wolf@ccrc.wustl.edu
[2] Departments of Computer Science and Electrical Engineering,
Washington University, St. Louis, MO, USA
jbf@ccrc.wustl.edu

Abstract. Demands for flexible processing have moved general-purpose processing into the data path of networks. With the development of System-On-a-Chip technology, it is possible to put a number of processors with memory and I/O components on a single ASIC. We present a performance model of such a system and show how the number of processors, cache sizes, and the tradeoffs between the use of on-chip SRAM and DRAM can be optimized in terms of computation per unit chip area for a given workload. Based on a telecommunications benchmark the results of such an optimization are presented and design tradeoffs for Systems-on-a-Chip are identified and discussed.

1 Introduction

Over the past decade there has been rapid growth in the need for reliable, robust, and high performance communications networks. This has been driven in large part by the demands of the Internet and general data communications. To adapt to new protocols, services, standards, and network applications, many modern routers are equipped with general purpose processing capabilities to handle (e.g., route and process) data traffic in software rather than dedicated hardware. Design of the network processors associated with such routers is a current and competitive area of computer architecture. This paper is aimed at examining certain tradeoffs associated with the design of these embedded network processors.

In the current router environment, single processor systems generally cannot meet network processing demands. This is due to the growing gap between link bandwidth and processor speed. Broadly speaking, with the advent of optical WDM links, packets are arriving faster than single processors can deal with them. However, since packet streams only have dependencies among packets of the same flow but none across different flows, processing can be distributed over several processors. That is, there is an inherent parallelism associated with the processing of independent packet flows. Thus, the problems of complex synchronization and inter-processor communications, typically encountered with parallelization arising from scientific applications, are not present. From a functional

and performance standpoint it is therefore reasonable to consider developing network processors as parallel machines.

There are a host of advantages associated with integrating multiple processing units, memory, and I/O components on a single chip and developing what is referred to as a SOC (System-On-a-Chip) network processor. Chief among them are the ability to achieve higher performance and, by using fewer chips, lower cost. Such implementations are however limited by the size of the chip that is feasible (for cost and technology reasons), the packaging technology that can be utilized (to achieve given pin requirements), and the power which can be dissipated (at a given frequency).Therefore, one important design decision for such multiprocessor chips is how many processors and how much associated cache should be placed on a single chip. This is important since, for a given chip size, more processors imply smaller caches and smaller caches lead to higher fault rates. High fault rates, in turn, impact performance and also the required off-chip memory bandwidth. Bandwidth requirements for off-chip memory access and network traffic I/O are yet another important design constraint. In this paper, we address these optimization issues. In particular, our contributions are:

- Development of a performance model for a general single chip multiprocessor oriented towards network processing, but applicable across a range of application domains. Such a model easily accommodates future technology changes that drive the design space.
- Exploration of the design tradeoffs available and development of optimal architecture configurations. In particular the model permits examination of the interactions between number of processors, size of on-chip caches, type of on-chip cache (SRAM, DRAM), number of off-chip memory channels, and characteristics of the application workload.
- Development of selected network processor design guidelines.

Two metrics are associated with the performance model presented. The first is processing power per unit chip area, and the second is the total processing power for a fixed size chip. Model evaluation is performed for a realistic network processor workload over a range of design parameters. The derived set of design curves can be used as guidelines for future network processor designs.

Section 2 that follows characterizes the overall system design in more detail. Section 3 covers the analysis of the optimization problem. Section 4 introduces the application workload that was used for the optimization results that are shown in Section 5. Section 6 summarizes the work and presents conclusions.

2 Multiple Processor Systems-On-a-Chip

For the remainder of the paper we focus on a single SOC architecture consisting of multiple independent processing engines (Figure 1). The memory hierarchy consists of on-chip, per-processor instruction and data cache, and shared off-chip memory. A cluster of processors shares a common memory channel for off-chip memory accesses. The I/O channel is used by the system controller/scheduler to send packets requiring processing to the individual processors.

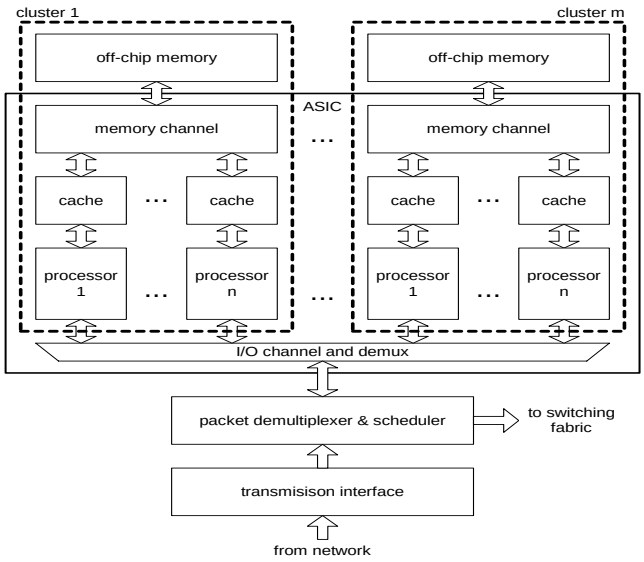

Fig. 1. Multiprocessor Router Port Outline.

Typically, a packet is first received and reassembled by the *Transmission Interface* on the input port of the router. The packet then enters a *Packet Demultiplexer* which uses packet header information to determine the flow to which the packet belongs. Based on this flow information the *Packet Demultiplexer* now decides what processing is required for the packet. The packet is then enqueued until a processor becomes available. When a processor becomes available, the packet and the flow information is sent over the I/O channel to one of the processors on the network processor chip. After processing has completed, the packet is returned to the *Packet Demultiplexer* and enqueued before being sent through the router switching fabric to its designated output port. A more detailed functional description of the above design can be found in [8]. Here, we consider the single chip design optimization problem associated with selection of the:

- Number of processors per cluster.
- Instruction and data cache size per processor.
- Cache memory technology (SRAM vs. DRAM).
- Bandwidth and load of the memory channels.
- ASIC size.
- Application workload.

Table 1 lists the parameters that are considered. The processors are assumed to be simple pipelined, general purpose RISC cores (e.g., MIPS [6], ARM [1], or PowerPC [5]). VLIW or superscalar processors are not considered since they require significantly more silicon real-estate than simple RISC cores. A study of

different multi-processor architectures [3] has shown that single chip multiprocessors are highly competitive with super-scalar and multithreaded processors. Also, super-scalar processors are optimized for workloads with few complex tasks rather than many simple and highly parallelized tasks that are found in the network processor environment.

Table 1. System Parameters.

Component	Symbol	Description
processor	clk_p	processor clock frequency
program	f_{load}	frequency of load instructions
	f_{store}	frequency of store instructions
	mi_c	i-cache miss probability for cache size c_i
	md_c	d-cache miss probability for cache size c_d
	$dirty_c$	prob. of dirty bit set in d-cache of size c_d
	$compl$	complexity (instr. per byte of packet)
caches	c_i	instruction cache size
	c_d	data cache size
	$linesize$	cache line size of i- and d-cache
	$t_{cache.dram}$	time for cache access (only DRAM)
off-chip memory	t_{mem}	time to access off-chip memory
memory channel	$width_{mchl}$	width of memory channel
	clk_{mchl}	memory channel clock frequency
	ρ	load on memory channel
I/O channel	$width_{io}$	width of I/O channel
	clk_{io}	clock rate of I/O channel
cluster	n	number of processors per cluster
ASIC	m	number of clusters and memory channels
	$s(x)$	actual size of component x, with $x \in \{ASIC, p, c_i, c_d, io, mchl\}$

3 Analysis

Given that we are interested in the amount of traffic the system can handle, we view the design problem as one of selecting the parameter values which maximize the throughput assuming chip area constraints, reasonable technology parameters, and the operational characteristics of a benchmark of network processing programs.[1]

Throughput in this environment corresponds to the number of packets that can be processed in a given time. This is determined by a combination of the instruction processing requirements of a given application (e.g., number of instructions necessary for routing table lookup, packet encoding, etc.), and the number of instructions that can be executed per second on the network processor. We assume that all packet processing tasks are performed in software on

[1] In this treatment we do not consider latency issues and assume that these requirements are met if the design can keep up with the incoming packet rate.

the RISC microprocessors. Thus, the throughput is proportional to the number of Instructions Per Second (IPS) that can be executed on the system. Given a typical RISC instruction set, network application benchmark characteristics, and various other parameters (e.g., CPU clock rate, cache miss times, etc.), an optimal system configuration, that maximizes IPS, can be determined.

3.1 Configurations

We begin by defining the fundamental chip area limitations for this system. The network processor chip size limits the number of processors, the amount of instruction and data cache per processor, and the number of memory channels that may be present. Let $s(ASIC)$ be the size of the network processor chip, $s(p_k)$, $s(c_{i_k})$, and $s(c_{d_k})$ respectively the sizes of a processor k, instruction cache c_{i_k}, and data cache c_{d_k}, and $s(mchl)$ and $s(io)$ the sizes of a memory channel and an I/O channel. With n processors per cluster and m clusters, all valid solutions must satisfy the following inequality:

$$s(io) + \sum_{k=1}^{n \cdot m} (s(p_k) + s(c_{i_k}) + s(c_{d_k})) + \sum_{k=1}^{m} s(mchl) \leq s(ASIC). \tag{1}$$

With identical processors, cache configurations, and I/O channels this becomes:

$$s(io) + m \cdot [s(mchl) + n \cdot (s(p) + s(c_i) + s(c_d))] \leq s(ASIC). \tag{2}$$

Further, we can assume that the best performance is achieved with a set of design parameters which result in an area as close to $s(ASIC)$ as possible. That is, we need to investigate only configurations that try to "fill" the available chip area. Another potential constraint concerns chip I/O pin limitations with a given packaging technology. We show later that this is not a significant constraint for the optimized systems considered.

3.2 Single Processor

Consider first the performance model for a single processor in terms of the number of instructions per second (IPS) that can executed by the processor. This metric is highly dependent on the processor architecture, however it does capture the effect of application instruction mix and memory hierarchy performance.

The number of executed instructions per second for a single processor, IPS_1, depends on the processor clock speed and the CPI:

$$IPS_1 = \frac{clk_p}{CPI} \tag{3}$$

In an ideal RISC processor, where there are no cache misses, branch mispredictions, or other hazards, all instructions can be pipelined without stalls and the CPI is 1. While in a realistic system the CPI increases with the occurrence of hazards, for this analysis, we only consider memory hazards since other hazards, like branch mispredictions, are relatively rare and cause only brief stalls

(1-2 cycles) in the short pipeline RISC processors considered here. This model constraint can be easily removed if greater accuracy is required. If SRAM is used as cache memory, a cache access can be done in one processor clock cycle and no stall cycles are introduced by cache hits. If DRAM is used for the instruction and data caches, then the basic pipeline clock cycle increases from 1 to $t_{cache.dram} \cdot clk_p$. Thus:

$$CPI = \begin{cases} 1 + p_{miss} \cdot penalty, & for\ SRAM \\ t_{cache.dram} \cdot clk_p + p_{miss} \cdot penalty, & for\ DRAM \end{cases} \quad (4)$$

where p_{miss} is the probability for an instruction cache miss or a data cache miss. The probability that a cache miss occurs, depends on the application being executed and the parameters associated with the caches. Using load and store frequencies and cache miss probabilities results in:

$$p_{miss} = mi_c + (f_{load} + f_{store}) \cdot md_c. \quad (5)$$

Note that Equation 5 considers only cache misses resulting from memory reads. Writes to memory, which are caused by replacing dirty cache lines, do not cause processor stalls. Assuming no contention for the memory channel, the miss penalty of a cache miss in turn depends on the memory access time and the time it takes to transfer a cache line over the memory bus (in processor clock cycles):

$$penalty = clk_p \cdot \left(t_{mem} + \frac{linesize}{width_{mchl} \cdot clk_{mchl}} \right). \quad (6)$$

With a cache miss, one cache line of size *linesize* is transferred over the memory channel. Additionally, if the replaced cache line was dirty, one cache line is written back to memory. The off-chip memory bandwidth generated by a single processor, $BW_{mchl,1}$, therefore depends on the number of instructions executed and how many off-chip accesses are generated. Thus:

$$BW_{mchl,1} = IPS_1 \cdot linesize \cdot (mi_c + (f_{load} + f_{store}) \cdot md_c \cdot (1 + dirty_c)). \quad (7)$$

The I/O bandwidth for a processor depends on the *complexity* of the application that is running. *Complexity* in the context of network processors is defined as the number of instructions that are executed per byte of packet data (header and payload). Applications with a high complexity require little I/O bandwidth, since more time is spent processing. Thus, the I/O bandwidth of a single processor, $BW_{io,1}$, is

$$BW_{io,1} = 2 \cdot \frac{IPS_1}{compl}. \quad (8)$$

The factor of 2 is present since every packet has to be sent first from the scheduler to the processor chip, and then later back out to the network. In the next section, this basic model is extended to the multiple processor situation.

3.3 Multiple Processors

Consider the case where multiple processors in a cluster share a common memory channel. Since the processors contend for the memory channel, it is necessary to account for the delay t_Q that is introduced by queuing memory requests. Equation 6 becomes:

$$penalty = clk_p \cdot \left(t_{mem} + t_Q + \frac{linesize}{width_{mchl} \cdot clk_{mchl}} \right). \tag{9}$$

To model the queuing delay, we approximate the distribution of memory requests due to cache misses by a exponential distribution. This reflects the bursty nature associated with memory locality processes.[2] Thus, the queuing system can be approximated by a M/D/1 queuing model. The deterministic service time corresponds to the duration of a cache line transfer over the memory channel. Given the load, ρ, on the memory channel, the average queue length for an M/D/1 queue can be expressed as:

$$N_Q = \frac{\rho^2}{2(1-\rho)}. \tag{10}$$

Multiplying by the time associated with a single non-blocked request, we obtain the average time for a request entering the system as:

$$t_Q = \frac{\rho^2}{2(1-\rho)} \cdot \frac{linesize}{width_{mchl} \cdot clk_{mchl}}. \tag{11}$$

The obtained cache miss penalty for the multiprocessor case (Equation 9) can now be used with Equation 4 to determine the CPI of a processor and Equation 3 then provides the number of instructions executed. If we know the number of processors, n, then multiplying by IPS_1 by n will result in the overall $IPS_{cluster}$. Using Equation 7 for the memory bandwidth generated by a single processor, n is the maximum number of processors that can be accommodated in a cluster without exceeding a selected load ρ:

$$n = \left\lfloor \frac{width_{mchl} \cdot clk_{mchl} \cdot \rho}{BW_{mchl,1}} \right\rfloor, \tag{12}$$

$$IPS_{cluster} = n \cdot IPS_1. \tag{13}$$

Knowing n, the size of such a cluster, $s(cluster)$, can be determined as the sum of all of its components (the I/O channel is not considered here, since it is shared over several clusters). Since n processors in a cluster share a single memory channel:

[2] Using a cache simulator, we measured the distribution of memory request interarrival times for the benchmark applications (Section 4). This was compared to an exponential distribution with the same mean. For $2kB$ instruction and data cache, the standard deviation of the measured interarrival times, on average, comes within a factor of 0.70 of the standard deviation of the exponential distribution.

$$s(cluster) = s(mchl) + n \cdot (s(p) + s(c_i) + s(c_d)). \tag{14}$$

Before turning to the optimization problem, we briefly discuss workloads that consist of multiple applications.

3.4 Multiple Applications

So far we have considered only a single program to be executed on the processors. A more realistic assumption is that there is a set of programs that make up the processor workload. The above analysis can easily be extended to accommodate such a workload notion.

Let the network processing workload W consist of l applications $a_1, a_2, ..., a_l$. Each application i is executed on a fraction q_i of the total data stream ($\sum q_i = 1$). The actual number of instructions that are executed by an application a_i depends on q_i and on its complexity, $compl_i$. Let r_i be the fraction of instructions executed that belong to application a_i.

$$r_i = \frac{q_i \cdot compl_i}{\sum_{k=1}^{l} q_k \cdot compl_k}, \; i = 1, ..., l \tag{15}$$

The fraction r_i determines the contribution of each application to memory accesses and associated pipeline stalls. The load and store frequencies $f_{load,i}$ and $f_{store,i}$ of each application a_i, the cache miss rates $m_{ic,i}$, $m_{dc,i}$, and the dirty bit probability $dirty_{c,i}$ are determined experimentally. The resulting average cache miss probability $p_{miss,W}$ for workload W is

$$p_{miss,W} = \sum_{i=1}^{l} r_i \cdot (mi_{c,i} + (f_{load,i} + f_{store,i}) \cdot md_{c,i}). \tag{16}$$

Similarly, the memory bandwidth $BW_{mchl,1,W}$ of a processor for workload W becomes:

$$\begin{aligned} BW_{mchl,1,W} = IPS_1 \cdot linesize \cdot \textstyle\sum_{i=1}^{l} r_i \cdot (mi_{c,i} + \\ (f_{load,i} + f_{store,i}) \cdot md_{c,i} \cdot (1 + dirty_{c,i})). \end{aligned} \tag{17}$$

The new definitions of $p_{miss,W}$ and $BW_{mchl,1,W}$ can be replaced in the above formulas to obtain n and IPS.

3.5 Optimization

The optimization process can be targeted either to a single cluster or to an entire chip containing multiple clusters:

- Processor cluster: The optimization of a processor cluster for different configurations helps to identify and understand basic design tradeoffs. It does not take into account global system components, like the I/O channel, and ASIC size constraints.
- Complete ASIC: The optimization of the complete system accounts for ASIC size and includes the I/O channel.

Based on the optimization goal, different optimization functions can be chosen. For the processor cluster, we define the number of instructions per second per area ($IPSA_{cluster}$) as:

$$IPSA_{cluster} = \frac{IPS}{size(cluster)}. \tag{18}$$

To find the maximum $IPSA_{cluster}$, theoretically any parameter shown in Table 1 can be varied. Practically, though, certain parameters, like $s(x)$ or $linesize$, are fixed and the optimization space can be limited to a smaller set of variables, such as clk_p, c_i, c_d, ρ, and whether the cache is implemented with on-chip SRAM or DRAM.

The complete ASIC optimization considers an integrated system consisting of several processor clusters on one chip. The number of clusters, m, is limited by the area constraint (Equation 2). The goal is to maximize the total number of instructions per second, IPS_{ASIC}, that can be executed on the ASIC.

$$IPS_{ASIC} = m \cdot IPS_{cluster}. \tag{19}$$

Due to the limited number of possible configurations, either optimization problem can be solved by exhaustive search over the configuration space.

4 Workload Definition

To properly evaluate and design network processors it is necessary to specify a workload that is typical of that environment. This has been done in the development of the benchmark CommBench [7]. Applications for CommBench were selected to include a balance between header-processing applications (HPA) and payload-processing applications (PPA). HPA processes only packet headers which generally makes them computationally less demanding than PPA that process all of the data in a packet.

For each application, the following properties have been measured experimentally: computational complexity, load and store instruction frequencies, instruction cache and data cache miss rate, and dirty bit probability. The complexity of an application can be obtained by measuring the number of instructions that are required to process a packet of a certain length (for header-processing applications, we assumed 64 byte packets):

$$compl = \frac{instructions\ executed}{packet\ size} \tag{20}$$

The cache properties of the benchmark applications were also measured to obtain $mi_{c,i}$, $md_{c,i}$, and $dirty_{c,i}$. This was done with the cache size ranging from $1kB$ to $1024kB$. For this purpose, a processor and cache simulator (Shade [2] and Dinero [4]) where used. A 2-way associative write-back cache with a linesize of 32 bytes was simulated. The cache miss rates were obtained such that cold cache misses were amortized over a long program run. Thus, they can be assumed to represent the steady-state miss rates of these applications.

We consider two workloads for the evaluation of our analysis: considered:

– Workload A - HPA: Header-processing applications.
– Workload B - PPA: Payload-processing applications.

These workloads are such that there is an equal distribution of processing requirements over all applications within each workload. Table 2 shows the aggregate complexity and *load* and *store* frequencies of the workloads. Note that the complexity of payload processing is significantly higher than for header processing. This is due to the fact that payload processing actually touches every byte of the packet payload and typically executes complex transcoding algorithms. Header processing on the other hand, typically only reads few header fields and does simple lookup and comparison operations. The aggregate cache miss rates for instruction and data cache are shown in Figure 2. Both workloads achieve instruction miss rates below 0.5% for cache sizes of $8kB$ or more. The data cache miss rate for workload A also drops below 0.5% for $8kB$. For workload B, though, the data cache miss rate only drops below 1% for $32kB$ or larger caches.

Table 2. Computational Complexity and Load and Store Frequencies of Workloads.

Workload	$compl_W$	$f_{load,W}$	$f_{store,W}$
A - HPA	9.1	0.2319	0.0650
B - PPA	249	0.1691	0.0595

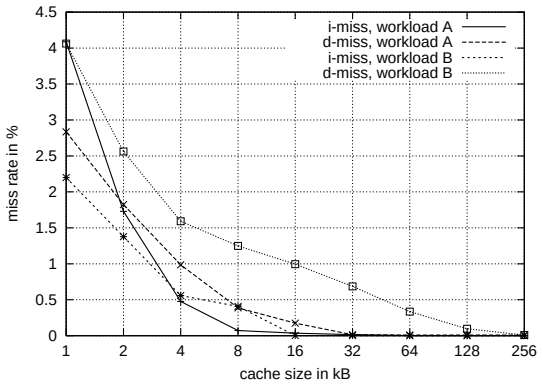

Fig. 2. Aggregate Cache Performance of Workloads.

5 Evaluation

For the optimization of the network processor we have to define a design space that reflects current ASIC technology. Table 3 shows the values, or ranges of values, of each system parameter considered. For the feature size of components, we assume .25μm technology.

Given the analysis of Section 3 and the workload and system properties of Section 4, the optimal configuration of a network processor can now be determined.

Table 3. System Parameters for Optimization.

Parameter	Value(s)	Parameter	Value(s)
clk_p	$50MHz \ldots 400MHz$	ρ	$0 \ldots 1$
c_i	$1kB \ldots 1024kB$	$width_{io}$	up to $64bit$
c_d	$1kB \ldots 1024kB$	clk_{io}	$200MHz$
$linesize$	$32byte$	$s(ASIC)$	$100mm^2 \ldots 400mm^2$
t_{mem}	$40ns \ldots 80ns$	$s(proc)$	$2mm^2$
$t_{cache.dram}$	$15ns$	$s(c_i),\ s(c_d)$	SRAM: $0.15mm^2$ per kB
$width_{mchl}$	$4bit \ldots 64bit$		DRAM: $0.015mm^2$ per kB
clk_{mchl}	$200MHz$	$s(mchl),\ s(io)$	$10mm^2 + width \cdot 0.25mm^2$

5.1 Cluster Optimization

This optimization looks only at the configuration of a cluster without considering ASIC chip size constraints or the I/O channel. Under these conditions, no area fragmentation occurs and design tradeoffs can be easily observed. For the two workloads, and the SRAM and DRAM configurations, we evaluate the effect of memory channel bandwidth and load, processor speed, and off-chip memory access time.

As base parameters, we use a memory channel bandwidth of $BW_{mchl} = 800MB/s$, a off-chip memory access time of $t_{mem} = 60ns$, and a processor clock speed of $clk_p = 400MHz$. Starting out with this configuration, we vary different parameters to see their effects on the overall system performance. Table 4 shows the optimal configuration for the base parameters.

For workload A, an $8kB$ instruction cache is sufficient to achieve very low instruction cache misses (see Figure 2). Workload B, requires a $16kB$ instruction cache. Since there is no "knee" in the data cache miss curve, the optimization results are $16kB$ and $32kB$ for data caches, which achieve less than 0.3% miss rate for workload A and less than 1% for workload B. Larger caches do not improve the miss rates significantly, but require much more chip area. The memory channel load for these configurations ranges from 69% to 79%. The number of processors per clusters is 6 and 16 when SRAM is present, and about 6 time larger, 40 and 91, when DRAM is present. The DRAM results stem from a combination of several effects: a) the processor speed is limited by the on-chip DRAM access time, b) the limited processor speed permits more processors to share a single memory channel, and c) DRAM takes about one tenth the area of SRAM. Despite the slower processing speed of DRAM configurations, they still achieve $50\% - 75\%$ of the $ISPA$ rating of the SRAM configurations.

One important observation is that increasing processor speed only affects SRAM cache configurations. This can be seen in Figure 3, where the $IPSA_{cluster}$ for different workloads and on-chip memory configurations is plotted over a set processor clock speeds. In the SRAM case, the total processing per area increases with faster processor clocks, since IPS_1 from Equation 3 increases. For DRAM cache, though, the effective processor speed is bound by the time it takes to access on-chip DRAM.

Table 4. Optimal configuration of cluster with base parameters of $BW_{mchl} = 800MB/s$, $t_{mem} = 60ns$, and $clk_p = 400MHz$.

Workload	On-chip Memory	n	c_i (kB)	c_d (kB)	ρ	$IPSA$ $(MIPS/mm^2)$
A - HPA	SRAM	16	8	16	0.72	50.31
B - PPA	SRAM	6	16	16	0.69	25.55
A - HPA	DRAM	91	8	16	0.78	25.20
B - PPA	DRAM	40	16	32	0.79	19.38

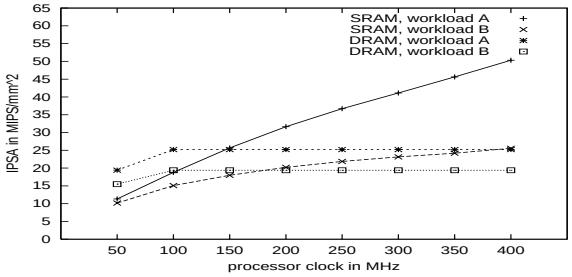

Fig. 3. Optimal $IPSA_{cluster}$ for different processor speeds ($BW_{mchl} = 800MB/s$ and $t_{mem} = 60ns$).

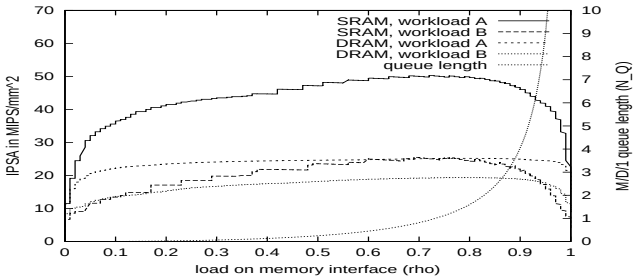

Fig. 4. Optimal $IPSA_{cluster}$ for different memory channel loads ρ ($BW_{mchl} = 800MB/s$, $t_{mem} = 60ns$, and $clk_p = 400MHz$).

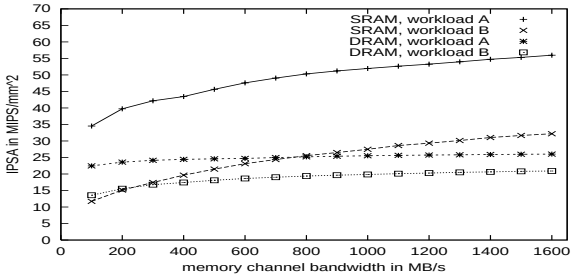

Fig. 5. Optimal $IPSA_{cluster}$ for different memory channel bandwidths ($t_{mem} = 60ns$ and $clk_p = 400MHz$).

The effect of the memory channel load, ρ, on the $IPSA_{cluster}$ is shown in Figure 4. Also shown on this figure is N_Q, the queue length associated with the M/D/1 model of the memory channel. The largest number of instructions executed can be achieved for memory channel loads of $\rho = 0.6 \ldots 0.85$, which corresponds to the region where the queue length is small (< 3). While smaller loads cause lower queuing delays, they also require more chip area per processor since fewer processors share the fixed size channel. Higher loads increase the queuing delay significantly, which in turn causes processors to stall very long on cache misses.

Figure 5 shows the $IPSA_{cluster}$ for different memory channel bandwidths. For SRAM configurations, a faster memory channel improves the $IPSA_{cluster}$ by about $20 MIPS/mm^2$, from $100 MB/s$ to $1600 MB/s$. This is due to the reduced transfer time for a cache line. These improvements are less significant for the DRAM configuration, since the processors operate at a much slower rate (bound by the on-chip DRAM access time) and the reduction in memory transfer time has less of an impact on the total CPI_{DRAM} (see Equation 4).

Different types of off-chip memories with different access times can also be used. The effect of the memory access time t_{mem} on the processing power per area is very limited. The total $IPSA_{cluster}$ decreases slightly for slower memories (2-5% for $t_{mem} = 80ns$ over $t_{mem} = 40ns$), but the memory access time is only a small component in the cache miss penalty (Equation 6). More important is the actual memory bandwidth and the load on the memory channel as shown in Figures 4 and 5.

5.2 ASIC Optimization

For the ASIC optimization, ASIC size constraints have to be considered as well as the I/O channel. To illustrate the optimization space, Figure 6 shows the optimal IPS_{ASIC} of a $400mm^2$ ASIC with workload A and SRAM caches. The memory channel bandwidth is $800 MB/s$, the processor clock speed is $400 MHz$, and the off-chip memory access time is $60ns$. One can see that there is a distinct optimum for $8kB$ instruction and $16kB$ data cache. Cache configurations that vary significantly from this optimum show a steep decline in overall performance. This emphasizes the importance of an optimally configured system.

Table 5. ASIC configurations with maximum processing power ($s(ASIC) = 400mm^2$).

Workload	Cache Type	m	n	c_i (kB)	c_d (kB)	ρ	t_{mem} (ns)	BW_{mchl} (GB/s)	clk_p (MHz)	I/O pins	IPS ($MIPS$)
A - HPA	SRAM	3	20	8	8	0.8	40	1.6	400	365	19700
B - PPA	SRAM	6	10	4	8	0.86	40	1.6	400	389	12600
A - HPA	DRAM	1	145	8	16	0.63	40	1.6	≥ 67[3]	148	9450
B - PPA	DRAM	1	64	16	16	0.92	40	1.6	≥ 67[3]	131	8050

[3] As explained in Section 5.1, the processor clock speed has no impact on the performance, as long as it is faster than the on-chip DRAM access time. Thus, any frequency above $67 MHz$ will achieve the same performance in this configuration.

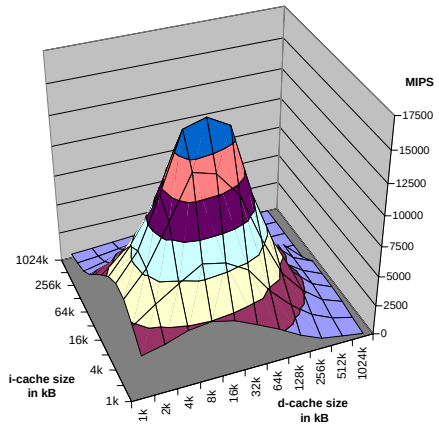

Fig. 6. Total processing capacity, IPS_{ASIC}, for different instruction and data cache sizes ($BW_{mchl} = 800MB/s$, $t_{mem} = 60ns$, $clk_p = 400MHz$ and $s(ASIC) = 400mm^2$).

The maximum IPS_{ASIC} found in any system configuration is shown in Table 5 for both workloads and cache technologies. It is not surprising that the optimum is achieved for the fastest technology parameters in all categories ($BW_{mchl} = 1.6GB/s$, $t_{mem} = 40ns$, and $clk_p = 400MHz$ for SRAM caches). The maximum processing capacity is almost $20000MIPS$ for an SRAM cache configuration with $8kB$ for data and $8kB$ for instructions. The DRAM cache configurations, again, achieve about half the performance of the SRAM caches. Note however, that the optimal DRAM configurations obtained do not take into account other factors which would likely make this design infeasible. For example, electrical bus loading would preclude having 145 processors associated with a single memory bus. Nevertheless, with improved DRAM implementations, the model will permit analysis of alternative configurations.

One important thing to note in Table 5 is that the maximum number of I/O pins (that is the number of data pins for the memory channels and the I/O channel) does not exceed 400. Even when adding pins that are necessary for signaling, control, and power, the total pin count does not go beyond current packaging technologies.

6 Summary and Conclusions

In this paper, we consider a multiprocessor System-on-a-Chip that is specialized for the telecommunications environment. Network traffic can be processed by special application software that executes on a set of processors contained on a single chip. The problem analyzed is that of determining the optimal number of processors, associated cache sizes, and memory channels that should be present in such a design given a set of defining parameters and constraints with the principal constraint being the total chip area available.

An analytical model of the system has been presented that reflects the computational power per area of a cluster and the total processing power of an ASIC. Using application statistics from a telecommunications benchmark, a workload was defined and used in the optimization process. Results for various cluster and ASIC configurations were presented and analyzed. The following key technology tradeoffs for System-on-a-Chip designs can be derived:

- The processor clock frequency has significant impact on configurations with on-chip SRAM caches. For on-chip DRAM caches, it does not improve the performance for clock rates higher that the memory access speed.
- Higher memory channel bandwidth improves both SRAM and DRAM configurations. The impact is larger for SRAM configurations. The optimal memory channel load for SRAM caches is in the range of 65% to 85% and for DRAM caches in the range of 70% to 92%.
- For the workload considered, the access delay of off-chip memory has little impact on the system performance.
- Optimal DRAM cache configurations achieve on average only half of the processing power of SRAM cache configurations, however, with current technologies, other implementation constraints likely make them a less desirable alternative than SRAM.
- Tradeoff trends are the same for both of the workloads considered. This indicates that they are independent of the particular workload for which the system is optimized.

These general observations along with the use of the model can be utilized to guide the design of network processors.

References

1. ARM Ltd. *ARM9E-S - Technical Reference Manual*, Dec. 1999. http://www.arm.com.
2. R. F. Cmelik and D. Keppel. Shade: A fast instruction-set simulator for execution profiling. In *Proc. of ACM SIGMETRICS*, Nashville, TN, May 1994.
3. P. Crowley, M. E. Fiuczynski, J.-L. Baer, and B. N. Bershad. Characterizing processor architectures for programmable network interfaces. In *Proc. of 2000 International Conference on Supercomputing*, Santa Fe, NM, May 2000.
4. J. Edler and M. D. Hill. *Dinero IV Trace-Driven Uniprocessor Cache Simulator*, 1998. http://www.neci.nj.nec.com/homepages/edler/d4/.
5. IBM Microelectronics Division. *The PowerPC 405^{TM} Core*, 1998. http://www.chips.ibm.com/products/powerpc/cores/405cr_wp.pdf.
6. MIPS Technologies, Inc. *JADE - Embedded MIPS Processor Core*, 1998. http://www.mips.com/products/Jade1030.pdf.
7. T. Wolf and M. A. Franklin. CommBench - a telecommunications benchmark for network processors. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 154–162, Austin, TX, Apr. 2000.
8. T. Wolf and J. S. Turner. Design issues for high performance active routers. *IEEE Journal on Selected Areas of Communication*, 19(3):404–409, Mar. 2001.