Maximizing Area Efficiency for Single-Chip Server Processors

Jaehyuk Huh Doug Burger Stephen W. Keckler
Computer Architecture and Technology Laboratory
Department of Computer Sciences
The University of Texas at Austin
cart@cs.utexas.edu — www.cs.utexas.edu/users/cart
IBM Mentor - Charles R. Moore, IBM Server Group

Abstract

In this paper, we study area budgets for future single-chip server processors. We balance the area of different processing core organizations and cache configurations to determine which organizations can maximize job throughput on a chip with large numbers of processors. We find that, in future server processors, out-of-order issue processing cores with small caches are the most area-efficient, which contradicts some designs that use small processors and large caches as the best way to obtain high throughput. We find that inequal scaling of processor transistors and pins will cause chip bandwidth to limit the number of processors that can be placed on a die. Finally, we show that the ideal configuration in a 35nm process incorporates 144 out-of-order processors, sustaining well over 130 instructions per cycle of execution.

1 Introduction

Innovations in silicon fabrication technologies continue to provide higher transistor densities and more transistors per chip with every generation. While increased transistor counts have largely been used to improve uniprocessor performance with wide out-of-order superscalar architectures, technology trends will present tremendous challenges for these architectures in the future. Clock rate improvements will be limited by pipeline depth constraints, and the increasing effect of wire delays will increase global chip communication delays [1]. To reduce the communication overheads, a large chip area could be partitioned into smaller relatively independent structures, such as in a chip multiprocessor (CMP), with each partition performing efficiently and suffering only limited effects from long global communication delays.

Workload characteristics are also changing in favor of those with increased concurrency. In addition to traditional parallel technical applications, commercial workloads such as database and web servers have become ubiquitous in high performance servers. The thread-level parallelism contained in these server applications make CMPs an attractive alternative to conventional microarchitectures. Furthermore, the reduced complexity and reusability of partitioned cores make CMPs particularly attractive in the face of increasing design cost and long wire delays.

However, limited pin bandwidth presents a substantial challenge to chip multiprocessors. Figure 1 shows the projected ratio between chip transistor capacity and pin count, according to the SIA Roadmap [19]. While pin count is increasing, the number of transistors is increasing at a much higher rate. For example, in a 35nm technology there are 34 times more transistors per pin than in a 180nm technology. If the increasing transistor count is used to implement multiple processors, the off-chip traffic will increase. In addition, the slow channel speeds and limited bandwidth of conventional DRAM architectures will make memory bandwidth the prime bottleneck for high-throughput server processors.

The primary trade-off in the design of a chip multiprocessor is between the use of the chip area and off-chip bandwidth. At the foundation of chip area allocation are the number of processing cores, the performance of each core, and the capacity of the on-chip memory system. Higher chip-level computation throughput can be obtained either by increasing the number of cores or by improving instruction throughput (instructions per clock or IPC) of each core. However, with a limited area budget a CMP architect cannot increase both without bound

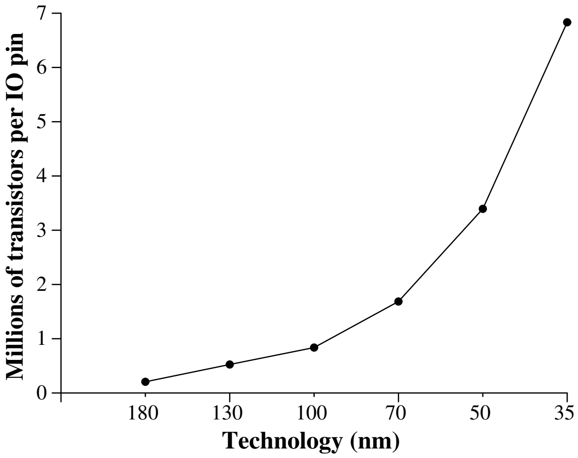

Figure 1: Transistors per IO pin versus technology.

and must determine the right balance between number of cores and core performance. Chip area must also be devoted to the on-chip memory hierarchy (level 1 and 2 caches) to reduce access latency and off-chip bandwidth demands. The organization of the off-chip memory channels is also important in achieving high performance. With a fixed number of pins, the designer must decide how many memory channels to implement and how wide to make each channel.

These design decisions have complex interactions. Increasing the number of processors requires that the cache capacity per processor decrease, which may result in an increased demand for off-chip bandwidth and no overall performance benefit. Likewise, using wide out-of-order processors requires more area per processor, but processor count may decrease and each processor may be able to tolerate additional memory latency.

The goal of this paper is to explore the trade-offs among different organizations of processors and memory channels. We focus on the chip area and memory bandwidth constraints, and show how processing cores and memory hierarchy should be organized to maximize the performance of the single-chip server processors. The initial results in this paper show that four-way issue out-of-order processors are more effective than simple in-order issue processors when enough off-chip bandwidth is available, that small caches provide maximum area efficiency, and that the disparate scaling trends of transistors and processor pins may join power dissipation as a major limit to the number of processing cores that a server chip may contain.

Section 2 describes the single-chip server architecture model and the basic assumptions of this study. Section 3 presents the experimental framework and methodology. Section 4 reports the experimental effect of each factor on overall performance. Section 5 discusses related work in chip multiprocessors and memory bandwidth studies. Finally, conclusions and future work are presented in Section 6.

2 Background

2.1 Chip-multiprocessor Architecture

A CMP consists of a group of processing cores, on-chip cache hierarchy, interconnection networks, and channels to external memory. The caches are either local caches, which are accessed only by one core, or shared caches. Current processors contain levels of local caches, which are connected either to shared caches or external memory. To provide a shared address space, the local caches are kept consistent by coherence protocols. In tightly coupled CMP architectures, cores can be connected to a shared cache without local caches, which enables fast communications among processing cores. Nayfeh defined the three principal CMP architectures, shared-L1,

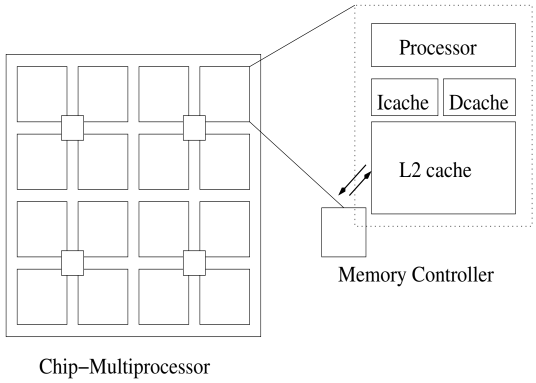

Figure 2: Chip-multiprocessor model.

shared-L2, and shared-memory, and evaluated them with parallel workloads [17]. They conclude that shared-L1 or shared-L2 architectures will be effective for fine-grained sharing of data.

However, as the transistor counts on a chip increase, a shared-L1 will not perform effectively as fast, private first-level caches, since cache bandwidth requirements and slow global wires will make large L1 caches shared by many processing cores too slow to be useful. Shared L2 caches will suffer from the same effects as the number of processing cores on the chip grow large. Current CMPs are addressing long access times of shared L2 caches by allowing non-uniform accesses. The Compaq Piranha [2] divides a L2 cache into eight sub-banks by address interleaving, and allows each core to access its local L2 bank faster than remote L2 banks.

While these assumptions are feasible in near-term server chips, we are examining long-term scalability of single-chip servers. Consequently, we assume that eventually, no cache will be globally shared, and that all L1 and L2 cache banks are private to their local processing cores. Further study on partial sharing of cache banks, which may be a feasible design point, is beyond scope of this paper.

Our CMP model thus has two levels of cache hierarchy, with L1 and L2 caches coupled to individual processing cores for scalability. Logical sharing and L2 bank coherence are still possible in our substrate, although we model neither. Finally, each L2 cache is connected to shared memory (See Figure 2), and accesses external DRAM through on-chip memory channels.

To provide enough memory bandwidth for a high-throughput CMP, we simulate fast, dedicated off-chip memory channels. Since the number of memory channels is limited by physical and economic constraints, the allocation of the finite bandwidth must be considered when designing cost-effective CMPs. Because of pin limitations, as the number of processors increases, private memory channels for processor/L2 cache will be infeasible. The effects of contention on shared channels will affect the ideal balance between cache and processor area allocations.

2.2 Maximizing Throughput

Since we focus on uniprocessor server workloads in this paper, we represent the performance of a server processor as being the aggregate performance of all the cores on a chip. For these workloads, two parameters—the number of cores (N_c), and the performance of each core (P_i)—are sufficient to estimate peak performance P_{cmp} of a server CMP.

$$P_{cmp} = \sum_{i=1}^{N_c} P_i$$

The performance of an individual core in a CMPs (P_i) is dependent on application characteristics such as available instruction level parallelism, cache behavior, and communication overhead among threads. For applications which spend significant portions of their execution time in communications and synchronization, parallel efficiency of the applications drops precipitously, and realized P_{cmp} will drop below peak P_{cmp}. However, in server applications, threads are initiated by independent clients, and they rely on relatively coarse-grained data sharing (or no sharing at all), thus resulting in high parallel efficiency.

To simplify our initial study on CMP designs, we focus on the ILP and cache behavior of applications, deferring a study of application communication and synchronization effects to future work. Our base assumption in this study is that processes are all independent of one another, which is the case in a multiprogrammed environment. The goal of our study is to find the best parameters for a server CMP if all the processes running on the CMP are independent.

2.3 Area Efficiency

Given a fixed area budget on a CMP, the fundamental goal is to balance the number of cores with the performance of each individual core, since investing more area in a core for either wider-issue execution or for larger caches will result in a smaller number of cores on the chip. We use the performance (IPC) per unit area (P/A) to find the optimal configuration of each core. If the area of a processor and its caches are represented by A_{proc} A_{il1}, A_{dl1}, and A_{l2} respectively, and we obtain the performance P of the processor with those caches, we can determine the area efficiency of the core with P/A:

$$P/A \;=\; P/(A_{proc} + A_{il1} + A_{dl1} + A_{l2})$$

To facilitate direct comparison of area trade-offs between processors and caches, a common metric is needed. Due to the difference of density between logic and memory units, comparing the transistor counts of processors and caches directly may be misleading. Furthermore, other overheads such as wires must be included in area estimates. To address these issues, we measure all structures in a single unit: the area required for one-byte of on-chip cache memory (cache byte equivalent). This one-byte metric incorporates all cache overhead such as tag area and decoding logic, in addition to the eight SRAM cells used to store a byte. The area of our processor models are estimated by comparing them the area of their on-chip caches. This processor area includes every on-chip structure except caches and memory controllers, and is expressed in cache byte equivalents.

3 Experimental Methodology

3.1 Processor and Cache Parameters

We use the SimpleScalar tool set, a widely used out-of-order processor simulator to simulate processing cores [3]. We evaluate two different processing core organizations to explore their area versus performance trade-offs. In Table 1, we list the features of the two processor organizations. The processor named P_{IN} is a simple 2-way in-order issue processor that is roughly comparable to the Alpha 21064 [16], The processor named P_{OUT} processor is a more aggressive, 4-way issue out-of-order processor comparable to the Alpha 21264 [13]. These simulated processors have different microarchitectures than the Alpha 21064 and Alpha 21264, but are intended to model processors of similar capabilities implemented with similar transistor budgets. We estimated the areas of the two processing cores by comparing the processing core area with the on-chip cache area from the die photos of the two Alpha processor models. We also validated the estimated areas against our technology-independent microarchitectural area model [10].

To simulate the effects of cache size on cache access latency, we used the ECacti tool to determine access latency as a function of cache capacity [18, 20]. Given the cache capacity, associativity, number of ports, and

	P_{IN}	P_{OUT}
Instruction Issue	in-order	out-of-order
Issue width	dual-issue	quad-issue
Instruction Window (entries)	16	64
Load/Store Queue (entries)	8	64
Branch predictor	bimodal (2K)	comb (1K+4K)
Number of Integer ALUs	2	4
Number of Floating-Point ALUs	1	2
Estimated Area (cache bytes)	50 KB	250 KB

Table 1: Processor model parameters.

L1 data cache		L2 cache	
16KB	2	128KB	4
32KB	2	256KB	5
64KB	2	512KB	7
		1MB	10

Table 2: Cache model access times at 100nm, measured in cycles.

number of data and address bits, ECacti finds the optimal cache configuration (minimal access time) by modeling a large number of alternative cache organizations. Table 2 shows an example of the the cache latencies obtained for a 100nm technology.

In addition to modeling the organizations to determine the latencies, we simulated non-blocking, write-back caches, and modeled bus contention at all levels. We assumed that the L1 instruction and data caches are two-way set associative with 64-byte blocks, and that the L2 caches are four-way set associative with 128-byte blocks. Finally, to reduce the experiment space, we fixed the L1 i-cache sizes to 32 KB, as our benchmarks have low instruction cache miss rates.

3.2 Memory Channel Model

Our memory simulator [9] models Direct Rambus memory channels [5] in detail, so that our CMP results incorporate aggressive, next-generation memory system technology. In this Rambus system, the data bus is clocked at 400 MHz, and data are transfered on both edges of the clock. A Rambus channel uses 30 pins for control and data signals, with a data width of 2 bytes. If more bandwidth is needed and pins are available, multiple Rambus channels may be used in concert to form a single, logically wider memory channel. The processor clock rate is set to four times that of the Rambus DRAM, and assumes that memory channel speeds will scale with processor clocks for future technologies. That ratio is consistent with a next-generation 1.6GHz processor incorporating the 400MHz DRDRAM parts.

3.3 Benchmarks

We simulated six benchmarks that were chosen to provide a wide range of memory system behavior. Five were drawn from the SPEC2000 suite: *gcc*, *mcf*, *equake*, *ammp*, and *parser*. The sixth is the Stream benchmark, a microbenchmark used to measure effective DRAM bandwidth in systems [15]. For each application, the first billion instructions of execution are skipped to avoid simulating benchmark initialization, and the subsequent 200 million instructions are simulated in detail.

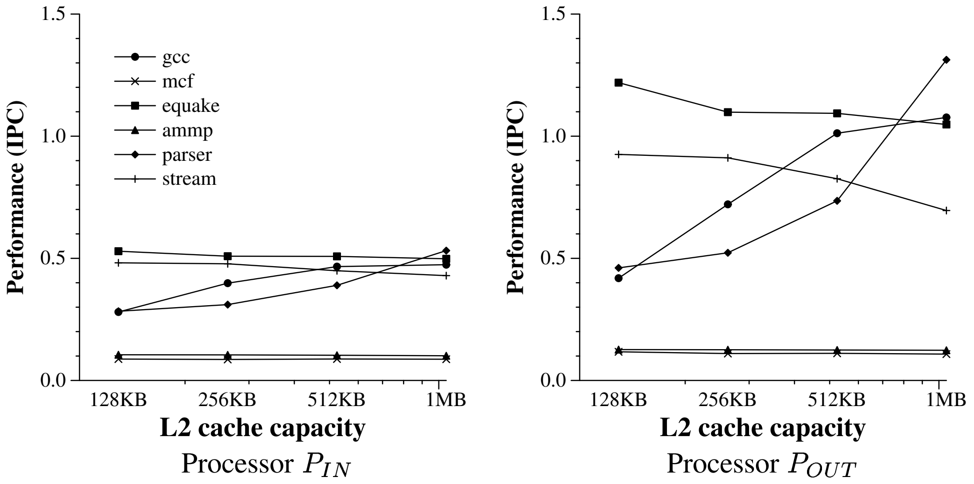

Figure 3: IPC versus L2 cache capacity.

4 Experimental Results

In this section, we first explore the performance improvements provided by larger caches for both of our processor models. We then consider the effect on performance per unit area when cache sizes grow. Finally, we vary the available memory bandwidth, to determine the sensitivity of the area-optimal cache and processor configuration to available channel bandwidth.

4.1 Cost-effective Processor and Cache Size

On-chip caches have recently come to dominate not only the transistor counts on-chip but the chip area as well. Dedicating area to cache memory has proven to be a simple way to improve performance reliably. However, it is unclear that large caches will the most efficient solution for server-oriented CMPs.

Figure 3 shows the effect of varying the L2 cache size on both the P_{IN} and P_{OUT} processors' performance, as measured in IPC. We assume 32KB split first-level caches, as well as 4-byte (dual) Rambus channels per processor. The L2 cache size has only a marginal effect on the in-order core, with *parser* and *gcc* showing factors of two improvement for 1MB over 128KB L2 caches, *stream* and *equake* showing slight degradations due to the longer access penalties, and two bandwidth-bound applications (*ammp* and *mcf*) showing no effect. The effects on benchmark performance are the same for the out-of-order core, but are much more strongly pronounced. If area were free, the 1MB cache with the out-of-order core would be the best solution. However, an eight-fold increase in cache area improves performance a maximum of 160%, implying that smaller caches are likely to be more area-efficient.

In Figure 4, we show the area efficiency of six different configurations for each benchmark. The configurations vary the L1 data cache size from 16KB to 64KB, for both P_{IN} and P_{OUT}. The metric we use for area efficiency is the IPC normalized to one megabyte of cache equivalent area. For example, an out-of-order core (250KB cache equivalent) with a 1.75MB cache that sustained 1.5 IPC would have an area efficiency of 0.75 IPC/MB.

The results show that the most area-efficient organizations are those with the smallest caches: 128KB L2 caches and 16KB L1 data caches. Given that doubling a cache capacity rarely produces a doubling in overall performance, this result is not unexpected. Surprisingly, P_{OUT} is shown to be more area-efficient than P_{IN}. Although the P_{OUT} core is five times as large as the P_{IN} core, the addition of the L1 and L2 caches make the area ratio between the two smaller than the performance ratio. The performance gained in tolerating long

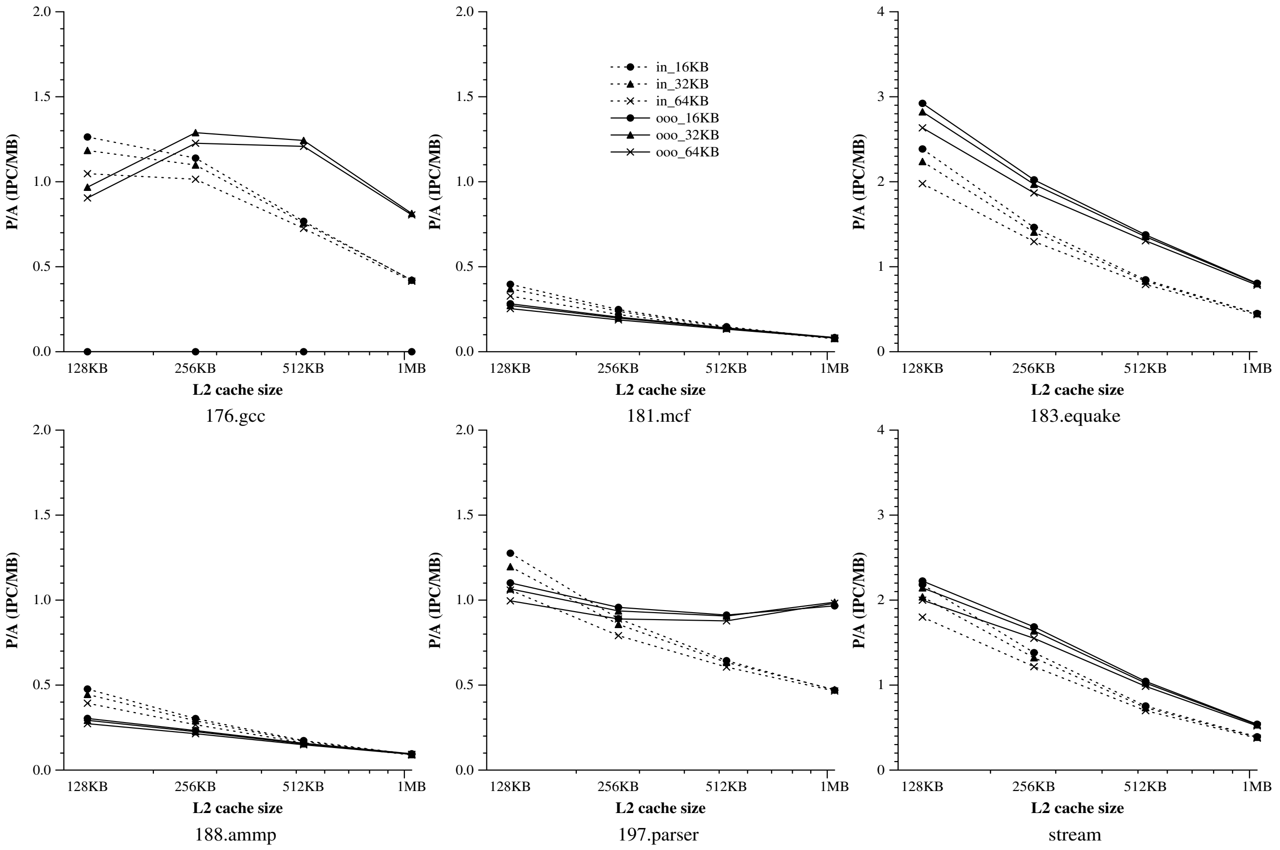

Figure 4: Area efficiency (IPC/MB cache) versus L2 cache capacity.

memory latencies and supporting wide issue outweighs the extra 200KB in cache equivalent area consumed by the P_{OUT} core. The P_{OUT} core with 16KB L1 and 128KB L2 is the most area-efficient in every case but one. The benchmark *gcc* achieves its best area efficiency with a 256KB L2 cache, since the gains in capturing more of the working set in the cache outweigh the extra 128KB of area consumed. Future work will explore a wider range of processor configurations, including wide in-order processors and cores with no L2 cache.

4.2 Effect of Memory Contention

Another constraint affecting the optimal organization of CMPs is chip-to-memory bandwidth. The memory bandwidth will become a more precious resources as technology advances [4]. First, the number of pins connecting a CMP with external memory increases more slowly than the number of transistors. While this effect is seen in single-processor chips, as CMPs become prevalent, the number of pins will eventually place a tighter bound on the number of CMP cores than will the availability of transistors.

Second, the times that the channels are idle are likely to be reduced as the bandwidth and latency of DRAM systems improve. The highly banked DRAM architecture, pipelining of bank accesses, and aggressive channel scheduling have enabled modern DRAMs to keep up with fast channels. Furthermore, as the improvement in processor clocks slows from the recent annual 40% to approximately 12% per year [1], the processor/memory latency gap will cease to grow, and may in fact shrink, putting more pressure on the memory channels themselves.

Our area efficiency results showed that in the absence of channel contention, increasing the cores per chip by implementing small caches is the best method of maximizing job throughput. As the number of processing

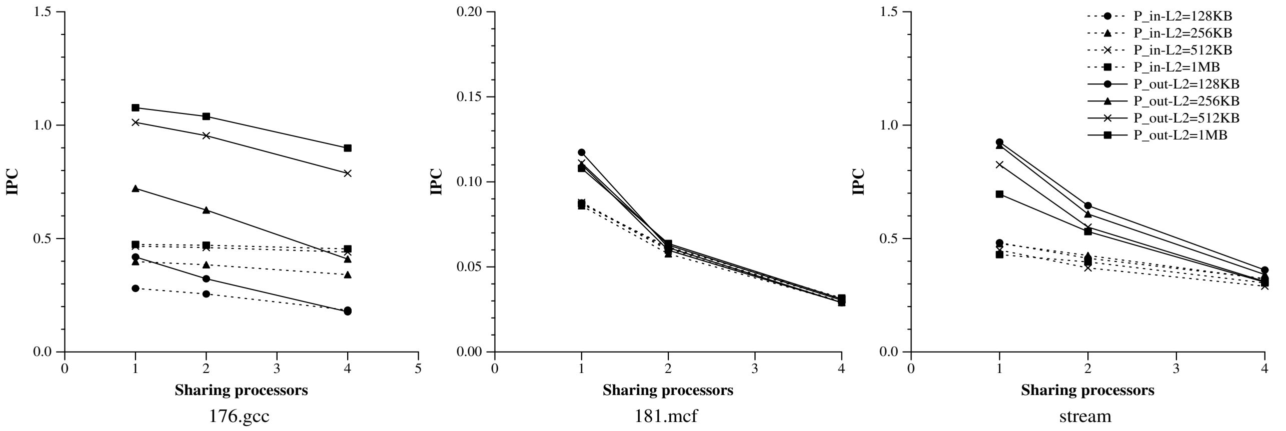

Figure 5: Processor IPC versus number of processors when sharing a memory channel.

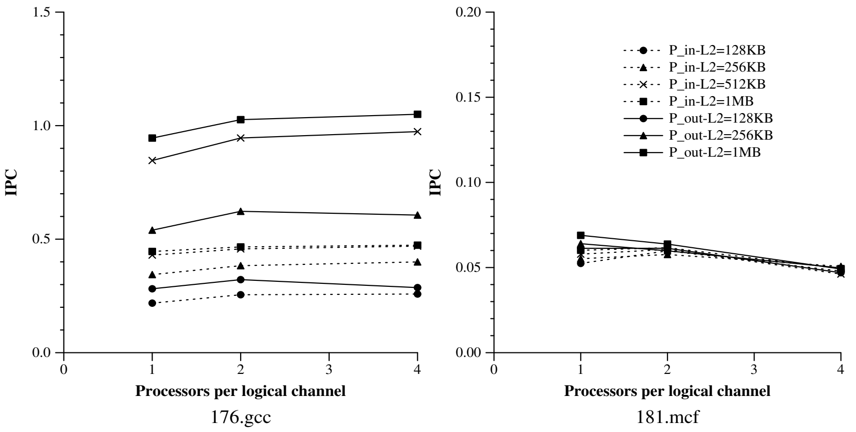

Figure 6: Processor IPC versus channel organization.

cores on a chip increases, and as memory bandwidth becomes a performance bottleneck, larger caches that reduce off-chip traffic will become more desirable. Figure 5 shows the impact of memory channel sharing on the performance of processing cores. The L1 d-cache size is fixed at 64KB, and Rambus channels are set at 4-byte wide.

As expected, adding processors sharing a single channel always reduces performance. The effect on P_{IN} cores is less severe since they have a lower channel utilization than P_{OUT}. The same amount of data is loaded across a longer execution time. For the two bandwidth-bound applications (*ammp* and *mcf*) in Figure 5, the performance of all configurations quickly converges to a level set by the available bandwidth, at which point the smallest configuration is clearly the most area efficient. With P_{OUT} processor cores, however, *gcc* experiences a larger degradation for the smaller caches as contention increases, indicating that limited bandwidth will make larger caches more area-efficient. If one or two cores share each channel, 256KB L2 caches are the most area efficient. If four cores share each channel, then cores with 512KB L2 caches become the most area efficient for gcc.

Figure 6 shows the results of an experiment that determines whether allocating pins to a single wide channel shared among cores or to narrower channels private to each core. Performance (IPC) is on the y-axis and the number of processors sharing a logical channel is on the x-axis. When the number of processors sharing the channel is doubled, so is the width of the channel, keeping the total number of pins constant. If the channels

Gate length	100nm	70nm	50nm	35nm
Cache equivalent area	7.6MB	15.5MB	30.5MB	62.0MB
Optimal number of cores	18	36	71	144

Table 3: Estimated area and number cores across fabrication technologies.

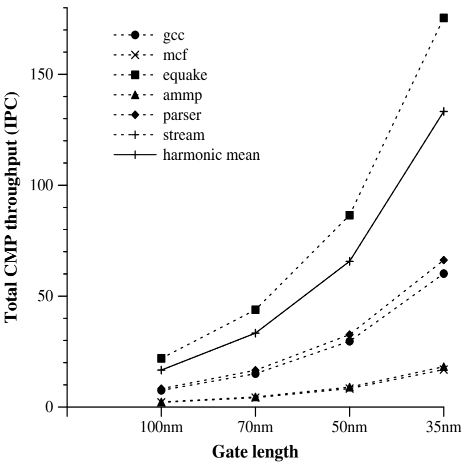

Figure 7: Application throughput across fabrication technologies.

are sparsely utilized, as in *gcc*, the P_{OUT} cores achieve higher performance, as a single core may benefit from having more bandwidth when the application enters a bursty phase. For *gcc* running on in-order cores, channel contention is sufficiently low so that performance remains unaffected. If all of the applications running have high bandwidth requirements, as in *mcf*, channel sharing increases contention, and causes application performance to suffer. This experiment does not consider the overhead of communicating with distributed channels, but doing so would likely favor private channels.

4.3 Estimating Total Throughput

In this section, we measure the maximal instruction throughput that can be obtained from a server processor, according to our most area-efficient configurations. Assuming one 4-byte channel per core and using the most area-efficient processor core and L2 organization from our results, we select the P_{OUT} core and 128KB L2 caches for performance estimation. To determine the amount of total cache that can be implemented on built on a chip, a $400mm^2$ chip area is normalized to λ^2, where λ is equal to one-half of a technology's minimum feature size. The λ metric is then combined with the area of one cache bit to compute the number of bits at each technology [10].

Table 3 shows the total cache equivalent area at each generation, as well as the number of P_{OUT} cores and their caches, that can fit on a chip. Finally, Figure 7 shows the total chip-level instruction throughput for each of the six benchmarks across four technologies. At 35nm, the harmonic mean of all six benchmarks exceeds 130 instructions per cycle, even though many of the benchmarks are bandwidth bound. These results indicate that if attention is paid to area efficiency, the number of useful cores, and consequently the total throughput, can be extremely high.

5 Related Work

Both research and development projects have recently designed and analyzed chip multiprocessor architectures. The Stanford Hydra project studied the memory hierarchy organization for a system consisting of four processing cores and on-chip caches [11]. They further proposed mechanisms for thread-level speculation to avoid sequential bottlenecks in parallel applications. Krishnan et al. have also examined mechanisms for exploiting thread-level parallelism from sequential binaries and speculative execution of extracted threads on CMPs [14]. The Compaq Piranha system will include eight light-weight single-issue in-order processors with 64KB data and instruction caches on a single chip [2]. The Piranha has a 1MB 8-way multi-bank shared L2 cache, and each L2 cache bank has a dedicated Rambus memory channel. The L1 and L2 caches are not inclusive as the total on-chip L1 cache capacity matches that of the L2. In the product realm, IBM has developed the Power4 multiprocessor chip targetted at commercial server workloads [7]. The Power4 chip consists of two out-of-order processing cores sharing an on-chip level-2 cache, with a total target clock rate exceeding 1 GHz.

Additional work has examined the efficiency of memory hierarchies and proposed mechanisms to balance processor and memory system performance. Jouppi et al. studied the optimal cache size for two level cache hierarchy of single-core processors [12]. Their research explored the trade-offs between miss rates and the latencies of various cache sizes. Farrens et al. addressed the area efficiency more directly by defining an Equivalent Transistor Count, which is an unit area for one bit of caches [8]. They determined efficient cache sizes for conventional single-core processors, and projected increased in area efficiency for multi-core systems over single core systems with large caches. However, their study did not model the differences between in-order and out-of-order processing cores. Burger et al. demonstrated the importance of memory bandwidth for application performance in future microarchitectures [4]. Cuppu et al. compared several modern DRAM architectures and examined the impact of latency and bandwidth on application performance, concluding that bandwidth is becoming a more critical resource [6].

6 Conclusion

In this paper, we have examined how to allocate chip area and pin bandwidth to processors and memory systems to maximize serial application throughput in a single-chip server processor. Increasing the chip area allocated to processing logic can increase the number of cores or the complexity of the cores (*i.e.* in-order to out-of-order) which can improve the theoretical throughput of the system. However, this strategy reduces the cache capacity per processor and can increase the overall demand for off-chip bandwidth. The number and width of channels to off-chip memory is limited by the fixed number of pins available on the CMP die. Increasing the number of channels reduces channel contention among processors, but may require channel width to be reduced. In our experiments, it is this off-chip bandwidth that will be the key constraint in designing future area-efficient CMPs.

In our experiments, we found that large caches are not efficiently used, and many cores with small per-processor caches are more beneficial than a few cores with large caches. Experiments comparing processor core models of differing complexities show that the memory latency tolerance of out-of-order processors enables them to be more area efficient than in-order processors that are one-fifth as large. In our experiments, the area efficiency of a single processor peaks with the smallest level-2 cache capacity (128KB) except for *gcc* which has slightly higher efficiency with a 256KB level-2 cache. In all cases, a 16KB level-1 cache is more area efficient than any larger capacity.

Our experiments in pin allocation indicate that bandwidth is critical to performance and throughput. When four processors share a single memory channel, the IPC of each processor may drop by up to 70%. With fixed pin capacity, the allocation between channel width and number becomes important. As transistors increase faster than processor pins, efficient channel design and management will become paramount.

As an initial study of cost-effective CMP designs, this paper addresses only a small part of the design space.

Future work will include a more detailed study of bandwidth allocation and an enhanced study of the interrelationship among processor count, on-chip memory capacity, large cross-chip wire delays, and off-chip bandwidth. In addition, while our existing sequential benchmarks may represent some CMP applications such as multiprogramming, many server applications share global data. Our future work will include analysis of server and technical workloads having a range of communication demands and overheads. Additionally, our work will include operating system effects and the interaction between processors and the rest of the system.

Acknowledgements

This work is supported by the National Science Foundation under CAREER awards CCR-9985109 and CCR-9984336, CISE Research Instrumentation grant EIA-9985991, University Partnership Awards from IBM, and a Shared University Research grant from IBM.

References

[1] V. Agarwal, M. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate versus ipc: The end of the road for conventional microarchitectures. In *The 27th Annual International Symposium on Computer Architecture*, pages 248–259, June 2000.

[2] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. Piranha: A scalable architecture based on single-chip multiprocessing. In *The 27th Annual International Symposium on Computer Architecture*, pages 282–293, June 2000.

[3] D. Burger and T. M. Austin. The SimpleScalar tool set version 2.0. Technical Report 1342, Computer Sciences Department, University of Wisconsin, June 1997.

[4] D. Burger, J. R. Goodman, and A. Kägi. Memory bandwidth limitations of future microprocessors. In *The 23th Annual International Symposium on Computer Architecture*, pages 78–89, May 1996.

[5] R. Crisp. Direct Rambus technology: The new main memory standard. *IEEE Micro*, 17(6):18–27, December 1997.

[6] V. Cuppu, B. Jacob, B. Davis, and T. Mudge. A performance comparison of contemporary DRAM architectures. In *The 26th Annual International Symposium on Computer Architecture*, pages 222–233, May 1999.

[7] K. Diefendorff. Power4 focuses on memory bandwidth: IBM confronts IA-64, says ISA not important. *Microprocessor Report*, 13(13), October 1999.

[8] M. Farrens, G. Tyson, and A. R. Pleszkun. A study of single-chip processor/cache organizations for large numbers of transistors. In *The 23th Annual International Symposium on Computer Architecture*, pages 338–347, April 1994.

[9] W. fen Lin, S. K. Reinhardt, and D. Burger. Reducing dram latencies with an integrated memory hierarchy design. In *The 7th Annual International Symposium on High-Performance Computer Architecture*, January 2001.

[10] S. Gupta, S. W. Keckler, and D. Burger. Technology independent area and delay estimates for microprocessor building blocks. Technical Report 2000-5, Department of Computer Sciences, University of Texas at Austin, April 2000.

[11] L. Hammond, B. A. Hubbert, M. Siu, M. K. Prabhu, M. Chen, and K. Olukotun. The stanford Hydra CMP. *IEEE Micro*, pages 71–84, December 2000.

[12] N. P. Jouppi and S. J. Wilton. Tradeoffs in two-level on-chip caching. In *The 23th Annual International Symposium on Computer Architecture*, pages 34–45, April 1994.

[13] R. Kessler. The Alpha 21264 microprocessor. *IEEE Micro*, 19(2):24–36, March/April 1999.

[14] V. Krishnan and J. Torrellas. A chip-multiprocessor architecture with speculative multithreading. *IEEE Transactions of Computers*, December 1999.

[15] J. D. McCalpin. Memory bandwidth and machine balance in current high performance computers. *IEEE Technical Committee on Computer Architecture Newsletter*, December 1995.

[16] E. McLellan. The Alpha AXP architecture and 21064 processor. *IEEE Micro*, 13(3):36–47, June 1993.

[17] B. A. Nayfeh, L. Hammond, and K. Olukotun. Evaluation of design alternatives for a multiprocessor microprocessor. In *The 23th Annual International Symposium on Computer Architecture*, pages 67–77, May 1996.

[18] G. Reinman and N. Jouppi. Extensions to cacti, 1999. Unpublished document.

[19] The national technology roadmap for semiconductors. Semiconductor Industry Association, 1999.

[20] S. J. Wilton and N. P. Jouppi. An enhanced access and cycle time model for on-chip caches. Technical Report 95/3, Digital Equipment Corporation, Western Research Laboratory, 1995.