Appears in the Proceedings of the 2001 International Conference on Parallel Architectures and Compilation Techniques (PACT 2001)

Exploring the Design Space of Future CMPs

Jaehyuk Huh Stephen W. Keckler Doug Burger
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

Abstract

In this paper, we study the space of chip multiprocessor (CMP) organizations. We compare the area and performance trade-offs for CMP implementations to determine how many processing cores future server CMPs should have, whether the cores should have in-order or out-of-order issue, and how big the per-processor on-chip caches should be. We find that, contrary to some conventional wisdom, out-of-order processing cores will maximize job throughput on future CMPs. As technology shrinks, limited off-chip bandwidth will begin to curtail the number of cores that can be effective on a single die. Current projections show that the transistor/signal pin ratio will increase by a factor of 45 between 180 and 35 nanometer technologies. That disparity will force increases in per-processor cache capacities as technology shrinks, from 128KB at 100nm, to 256KB at 70nm, and to 1MB at 50 and 35nm, reducing the number of cores that would otherwise be possible.

1 Introduction

Chip multiprocessors (CMP) designs are a promising approach for increasing job throughput in servers. The first CMPs are starting to appear in the commercial sphere, notably IBM's Power4 design, which has two processors. The Compaq Piranha research effort, though not implemented for a commercial product, evaluates using CMPs with many small cores for server workloads. It is likely that future CMPs will have considerably larger numbers of processors than today, for two reasons [1]. First, the superscalar paradigm is reaching diminishing returns, particularly as clock scaling will soon slow precipitously. Second, global wire delays will limit the area of the chip that is useful for a single conventional processing core. Since a single processing core will be unable to use the bulk of the chip real estate, the additional transistors will likely be used for additional cores.

In this paper, we present the first study of how CMP designs will evolve as CMOS technology is scaled to ultra-small (35 nanometer) devices. For each technology generation over the next decade, we determine the CMP organizations that maximize total chip performance, which is equivalent to job throughput in this study. We consider the following factors:

- *Processor organization*: Whether powerful out-of-order issue processors, or smaller, more numerous in-order processors provide superior throughput.
- *Cache hierarchy*: The amount of cache memory per processor that results in maximal throughput. The ideal capacity is a function of processor organization, memory latency, and available off-chip bandwidth.
- *Off-chip bandwidth*: Finite bandwidth limits the number of cores that can be placed on a chip, forcing more area to be devoted to on-chip caches to reduce bandwidth demands.
- *Application characteristics*: Applications with different access patterns require different CMP designs to attain the best throughput. Different applications display varying sensitivities to L2 cache capacity, resulting in widely varying bandwidth demands.

These constraints have complex interactions. More powerful processors place a heavier individual load on the off-chip memory channels, but smaller, more numerous processors may result in a heavier aggregate bandwidth load. Larger caches reduce the number of off-chip accesses, permitting more processors to share a fixed bandwidth, but the larger caches consume significant area, resulting in room for fewer processing cores. In this paper, we study the relative costs in area versus the associated performance gains, showing the organizations that maximize performance per unit area for future technology generations. Since our focus is on performance bounds, we do not consider power limitations in this study.

Our results show that the number of processing cores that maximize total job throughput does indeed grow large with future technologies, reaching 18, 28, 23, and 47 processors at 100, 70, 50, and 35 nanometers. However, we find that the useful number of processors will be limited by off-chip bandwidth, because the number of transistors is predicted to increase much faster than the number of signaling pins [22]. We also find that, when applications are not bandwidth bound, the design that supports maximal chip throughput uses powerful out-of-order cores with small (128KB) level-two caches per core at 100 nanometers. At smaller feature sizes, as more processors are forced to share memory channels due to restricted pin counts, many applications begin to be limited by memory bandwidth. In that case, larger per-processor caches are necessary to reduce the off-chip

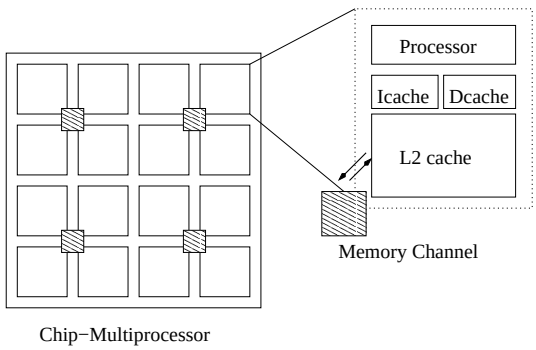


Figure 1. Chip-multiprocessor model.

load, resulting in ideal designs of 256KB per-processor L2 caches at 70nm, and 1MB per-processor caches being ideal at 50nm and 35nm.

In Section 2, we describe the CMP organization that we use for our experiments, as well as the area model and pin count extrapolations that we use for determining maximal performance/area. In Section 3, we explore how shrinking technology and different cache and memory channel organizations affect the performance of applications running on single processing cores. In Section 4, we couple the performance results with an area analysis to determine the highest performance (throughput) per unit area at each technology. In Section 5, we discuss the relevant characteristics of commercial server workloads, which we do not model in this study. In Sections 6 and 7, we survey related work and conclude.

2 Technology models for evaluating CMP alternatives

In this paper, we focus on throughput-oriented workloads with no sharing of data among tasks to evaluate the area efficiency of chip-multiprocessors. As shown in Figure 1, our CMP model has two levels of cache hierarchy, with L1 and L2 caches coupled to individual processing cores for scalability. An alternative to this cache hierarchy is a physically shared L2 cache. However, the performance of a large monolithic L2 cache shared by a number of processors will sharply diminish with advances in fabrication processes and increases in clock rates, due to large cache bandwidth requirements and slow global wires. Logical L2 cache sharing and L2 bank coherence are possible in our designs, although the applications presented in this paper have no inter-processor sharing. Each L2 cache is connected to the off-chip DRAM through a set of distributed 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. Thus our models account for time-multiplexing of the memory channels, and we investigate effects of channel contention on the ideal balance between cache and processor area allocations.

2.1 Area models

An analysis of area efficiency requires accurate models of processing cores and caches of varying capacities. We have derived a set of technology-independent area models empirically, by measuring die photographs of commercial microprocessors and normalizing the results for feature size [13]. To enable simple area trade-offs between processor core areas and cache bank areas, the model expresses all area in terms of cache byte equivalent area (CBE), which is the unit area for one byte of cache, similar to the Equivalent Cache Transistor metric of Farrens *et al.* [12]. We use a metric expressed in bytes for greater ease in reasoning about processor and cache area trade-offs. The CBE includes the amortized overheads for tags, decoders, and wires, in addition to the 8 SRAM cells.

For our processor model, we considered in-order and out-of-order issue processors ranging from 2-way to 8-way issue widths. In Table 1, we show the harmonic means of IPCs of our benchmarks listed in Section 3, for each model with varying L2 cache size. The number of ALUs are scaled with the issue width. For in-order cores, issue width has little impact on the performance, but out-of-order cores have significant performance improvement from 2 to 4 way issue cores. When the performance per unit area is considered, we found 2-way in-order and 4-way out-of-order processors are the most area-efficient models, and chose them as our processing core models. Table 2 shows the complete configuration of the two processor models used in this paper. P_{IN} is a simple 2-way in-order issue processor that is roughly comparable to the Alpha 21064 [20]. P_{OUT} processor is a more aggressive, 4-way issue out-of-order processor comparable to the Alpha 21264 [17]. 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. The results of this model show that the core area of P_{OUT} is five times larger than that of P_{IN}.

This paper assumes a large but fixed sized die of $400mm^2$ ($20mm \times 20mm$) across all of the technologies. With smaller feature sizes, the available area for cache banks and processing cores increases. Table 3 displays die area in terms of the cache-byte-equivalents (CBE), and for reference, λ^2 where λ is equal to one half of the feature size. The P_{IN} and P_{OUT} columns show how many of each type of processor with 32KB separate L1 instruction and data caches could be implemented on the chip if no L2 cache

	L2 cache size	2-way	4-way	8-way
In-order	128KB	0.20	0.21	0.21
	256KB	0.23	0.24	0.25
	512KB	0.24	0.25	0.25
	1MB	0.27	0.28	0.29
Out-of-order	128KB	0.26	0.31	0.33
	256KB	0.31	0.38	0.40
	512KB	0.32	0.39	0.41
	1MB	0.38	0.47	0.50

Table 1. Harmonic means of IPCs for six processor models.

	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)	16	64
Branch predictor	bimodal (2K)	2 level (16K)
Number of integer ALUs	2	4
Number of floating-point ALUs	1	2
Estimated core area (CBE)	50 KB	250 KB
L1 Instruction cache	32 KB	32 KB
L1 Data cache	32 KB	32 KB
Total area (core + I/D caches)	114 KB	314 KB

Table 2. Processor model parameters.

area were required. The primary goal of this paper is to determine the best balance between per-processor cache area, area consumed by different processor organizations, and the number of cores on a single die.

2.2 I/O pin bandwidth

While increasing transistor budgets can accommodate large numbers of processing cores on a single chip, the communication between the chip and the rest of system is both critical for performance and expensive to scale. The number of signal I/O pins built on a single chip is limited by physical technology and does not scale with the number of transistors. Figure 2 shows the projected ratio between chip transistor capacity and signal pin count, according to the SIA Roadmap, along with the absolute number of signal pins projected to be available in each technology [22].

Gate length	CBE (Megabytes)	λ^2 area	P_{IN}	P_{OUT}
100nm	7.6	1.60e+11	68	24
70nm	15.5	3.26e+11	139	50
50nm	30.5	6.40e+11	273	99
35nm	61.9	1.30e+12	556	201

Table 3. Total Chip Area.

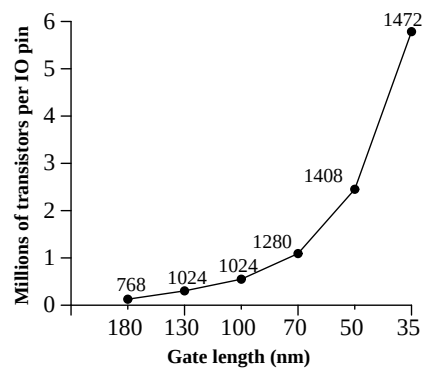


Figure 2. Transistors per IO pin.

While pin count is increasing, the number of transistors is increasing at a much higher rate. For example, in a 35nm technology there will be 45 times more transistors per pin than in a 180nm technology.

Another factor limiting off-chip communication is that, to date, I/O signaling speeds have not increased at the same rate as processor clock rates. It is common today to find a 1 GHz processor connected to memory through a 133MHz back-side bus. Even though active research aims to improve pin bandwidth by substantially increasing the pin transfer rates into the Gigabit per second regime [9, 11, 26], the disparity between the computation capacity and off-chip bandwidth will persist for the foreseeable future. For our experiments, we scale the chip pin density according to the SIA projections for signal pin density at a fixed $400mm^2$ die size. We scale the pin speeds linearly with technology at one-half the speed of the processor clock. This ratio is consistent with a next-generation 1.6GHz processor incorporating the 400MHz DRDRAM parts with dual-edge signaling, for an effective 800MHz data transmission rate.

2.3 Maximizing throughput

In a CMP, the performance on server workloads can be defined as the aggregate performance of all the cores on the chip. For these workloads, two parameters—the number of cores (N_c), and the performance of each core (P_i)—are necessary 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 CMP (P_i) is dependent on application characteristics such as available instruction level parallelism, cache behavior, and communication overhead among threads. For applications that spend significant portions of their execution time in communication and synchronization, parallel efficiency of the applications drops precipitously, and realized P_{cmp} will drop below

peak P_{cmp}. However, in many 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 serial applications, deferring a study of application communication and synchronization effects to future work. Our base assumption in this study is that all processes are independent of one another, which is the case in a multiprogrammed environment. The metric of performance in this paper is total throughput, measured in instructions per clock (IPC). Given a fixed die size, this metric is equivalent to an area efficiency metric. The optimization goal is to balance the number of cores with the performance and bandwidth demands of individual cores.

3 Application characteristics

The best allocation of processor area, cache area, and bandwidth depends on the the characteristics of the applications in the workload. This section characterizes the applications in this study based on their resource demands. We chose ten applications from the SPEC2000 benchmark suite and the *sphinx* speech recognition application to provide a wide range of memory system behavior. The SPEC2000 applications include *mesa*, *mgrid*, *equake*, *gcc*, *ammp*, *vpr*, *parser*, *perlbmk*, *art* and *mcf*. The experimental results show that the applications can be categorized by the following criteria:

- Processor-bound: applications whose working sets are captured easily in the L2 cache, who require few external DRAM accesses, and as a result are largely insensitive to cache capacity and bandwidth restrictions. *Mesa*, *mgrid*, and *equake* are in this class.
- Cache-sensitive: applications whose performance is limited by L2 cache capacity, as larger caches capture increasing fractions of the working sets. *Gcc*, *ammp*, *vpr*, *parser*, and *perlbmk* are in this class.
- Bandwidth-bound: applications whose performance is limited strictly by the rate that data can be moved between the processor and the DRAM. The working sets of the applications are much larger than L2 cache size, or there is little locality in the access patterns. *Art*, *mcf*, and *sphinx* are in this class.

Applications are not bound to one class or another; they move among these three domains as the processor, cache, and bandwidth capacities are modulated. To help characterize the memory behavior of the applications, we use the metric of DRAM references per thousand instructions. A DRAM reference results directly from an L2 cache miss or writeback. Consequently, this metric follows directly from the characteristics of the application, the L2 cache capacity, and as shown in Section 3.4, the number of processors in the CMP, due to sharing of memory channels by multiple processing cores.

3.1 Experimental methodology

We measure instruction throughput and memory behavior using the SimpleScalar tool set [5]. We configured SimpleScalar to model both the in-order and out-of-order processors, P_{IN} and P_{OUT}, described in Table 2. We further modified SimpleScalar for the chip-multiprocessor experiments to run multiple copies of the same application with varying numbers of memory channels and sharing of the channels among the processors. The memory system simulates non-blocking, write-back caches, and bus contention at all levels. The L1 instruction and data caches are two-way set associative with 64-byte blocks, and the L2 caches are four-way set associative with 128-byte blocks. To focus more directly on the larger L2 caches, L1 instruction and data caches are fixed at 32KB. For our benchmarks, the applications show little performance improvement with larger L1 caches, due to projected increase in access delays at smaller technologies. We therefore used the smallest of these equivalently performing cache organizations, since it was the most area efficient.

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 [21, 25]. Given the cache capacity, associativity, number of ports, and number of data and address bits, ECacti finds the best cache configuration (minimal access time) by modeling a large number of alternative cache organizations. In this section, we use only the parameters of a 70nm process. In this technology, the cache hit latencies of 128KB, 256KB, 512KB, and 1MB are 4, 5, 7, and 9 cycles, respectively.

To account for aggressive, next-generation memory system technology, the DRAM portion of our simulator models Direct Rambus memory channels in detail [7]. The data bus is clocked at 400 MHz, and data are transferred 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. We use two Rambus channels for our memory channel, resulting in a total of 60 pins per channel with a data width of 4 bytes. As mentioned in Section 2, the Rambus DRAM clock rate is set to effectively one-half (one-quarter clock with dual-edge transition) that of the processor, and assumes that memory channel speeds will scale with processor clocks for future technologies.

For each application, the first five billion instructions of

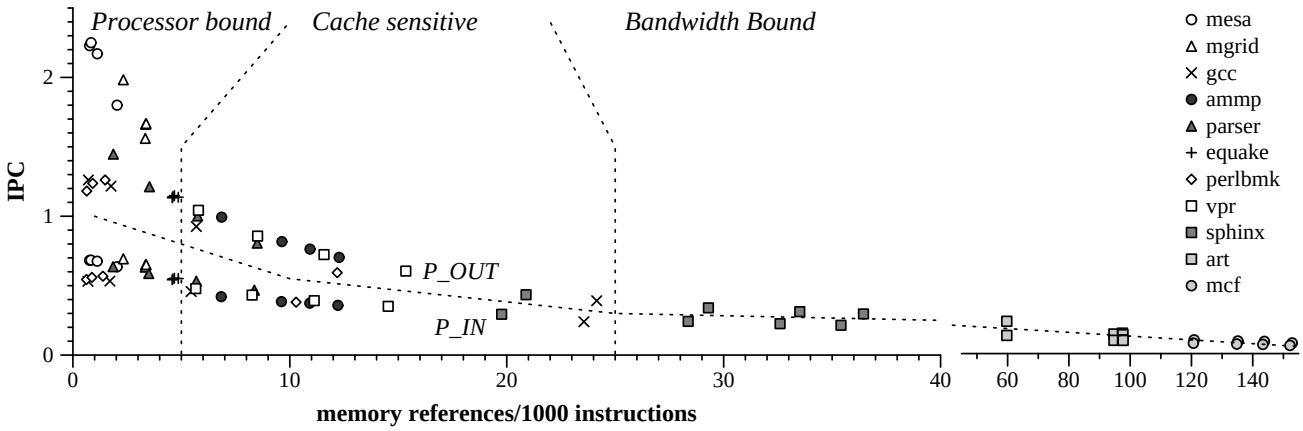


Figure 3. IPC versus rate of DRAM accesses.

execution are skipped to avoid simulating benchmark initialization, and the subsequent 200 million instructions are simulated in detail.

3.2 Application resource demands

To investigate the uniprocessor memory requirements of the applications, we varied the processor model and L2 cache capacity to modulate the DRAM reference frequency. The resulting instruction throughput is shown in Figure 3, as a function of reference frequency. For each application, a family of points is plotted corresponding to the two processor models (P_{IN} and P_{OUT}) and L2 cache capacities ranging from 128KB to 1MB. The general behavior for all of the applications is an increase in DRAM reference frequency as cache capacity is decreased, resulting in a reduction in IPC. Unsurprisingly, the IPC for the out-of-order processor uniformly exceeds that of the in-order processor, although the two organizations converge as the applications become bandwidth bound, with DRAM reference frequencies greater than 25 per 1000 instructions. We note that the benchmarks exhibiting more than 4 references per 1000 instructions show remarkably similar performance as a function of DRAM access rate. The applications can be divided into the three categories based on their position along the x-axis:

- Processor-bound benchmarks: The applications *mesa* and *mgrid* have few DRAM references per instruction. The IPC for these programs is high, particularly for P_{OUT}. The IPC, as well as the DRAM reference frequency, is largely insensitive to cache capacity. *Equake* exhibits similar behavior, even though its DRAM reference frequency is much larger than *mesa* and *mgrid*. Thus processor-bound applications show relatively high IPC and have working sets small enough to fit into moderately sized L2 caches.

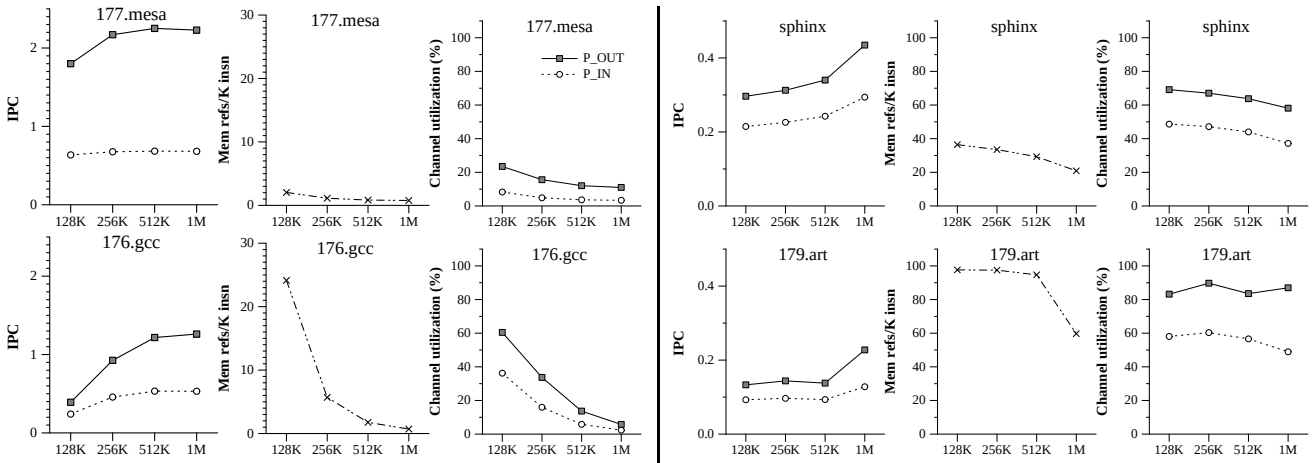
- Cache-sensitive benchmarks: The DRAM reference frequency and performance of *gcc*, *ammp*, *parser*, *perlbmk*, and *vpr* are much more dependent upon the L2 cache capacity. As cache sizes increase, the memory references tend to drop, making these applications appear to be processor-bound, particularly when the cache becomes large enough to hold the current working set. With smaller cache capacities, reference frequency increases and IPC drops substantially.

- Bandwidth-bound benchmarks: *sphinx*, *art* and *mcf* place enormous bandwidth demands on the off-chip interconnect. Even though large L2 caches reduce DRAM reference frequency somewhat, the effective lack of a working set results in low IPC even with the largest 1MB L2 caches. Because the L2 cache hit rate is so low, performance is directly proportional to the available bandwidth.

3.3 Processor organization and cache size

As shown in Figure 3, uniprocessor performance depends both on the processor organization and cache capacity. However, the effectiveness of increased cache capacity and out-of-order processors is limited by the bandwidth demands of the applications. To display these characteristics more clearly, Figure 4 shows the IPC, DRAM access frequency, and memory channel utilization as a function of cache capacity. Four applications are shown: *mesa* (processor bound), *gcc* (cache sensitive), and *sphinx* and *art* (bandwidth bound).

From this figure, we note the following points. First, the gap between the P_{IN} and P_{OUT} configurations in columns (a) depends on the memory demands of the benchmark. The



(a) Performance (b) DRAM refs (c) Channel util. (a) Performance (b) DRAM refs (c) Channel util.

Figure 4. Effect of varying L2 cache size.

gap is the largest for the processor-bound benchmark (*mesa*) , indicating that out-of-order cores will be more area efficient for that category. For the other benchmark (*gcc*), the performance of the out-of-order and in-order cores converges, as cache size drops and more frequent requests are made to memory.

Second, the data in columns (b) indicate that larger caches cause sharp reductions in L2 misses for the cache-sensitive benchmarks (and for *art* when the cache grows sufficiently large).

Finally, in columns (c) the data show that the out-of-order cores place heavier demand on the channel utilization. That demand results from the P_{OUT} cores moving the same quantity of data across the wires in a shorter time. We also note that the Rambus channels saturate at approximately 80% utilization, due to finite bandwidth on the command buses.

Several working sets are clearly visible in these data. When the L2 cache is increased from 256KB to 512KB, *gcc* shifts from the cache-sensitive category to being processor bound. The miss rate for *art* drops significantly when the L2 cache is increased to 1MB. However, even with that drop, *art* is still bandwidth bound, with over 50 DRAM accesses per 1000 instructions.

We define processor bound as having fewer than 5 off-chip accesses per 1000 instructions, and bandwidth bound as greater than 25. With that definition, it is clear that only two of the benchmarks shown here could tolerate any significant channel sharing: *gcc* for caches greater than 256KB, and *mesa* for any of the cache sizes that we measured.

3.4 Channel sharing

Channel sharing arises when multiple processes are executing simultaneously on different processors. Figure 5 plots the aggregate IPC seen by a number of processors sharing a single channel. The data show that the processor-bound job *mesa* exhibits good scaling of throughput with increased numbers of channel sharers, except for those experiments with the smallest (128KB) caches. *Gcc* scales or saturates, depending on whether the cache is large enough to hold its distinct, 400KB working set. The bandwidth-bound jobs *sphinx* and *art* show no improvement as more jobs are added, since their bandwidth is the critical resource and already saturated at one job. We note that again, the performance of the in-order and out-of-order cores converge as applications become bandwidth bound. Once too many processors are sharing a channel, adding more processors no longer improves throughput; that area would be better spent increasing the sizes of the caches and reducing the load on the channel. It is exactly that area/performance trade-off that we evaluate in the subsequent section. The utilization of the channel matches the throughput scaling; when the channel starts to reach saturation, throughput levels off.

In Figure 6, we plot the utilization of the channel that is shared by one to eight processing cores in *gcc*. When the channel in *gcc* becomes saturated for 128KB cache, utilization drops as more sharers are added. This counterintuitive result occurs because of decreased row buffer locality in the DRDRAM banks. Increased row misses cause gaps in the Rambus command bus schedule, which manifest as slightly lower data channel utilization.

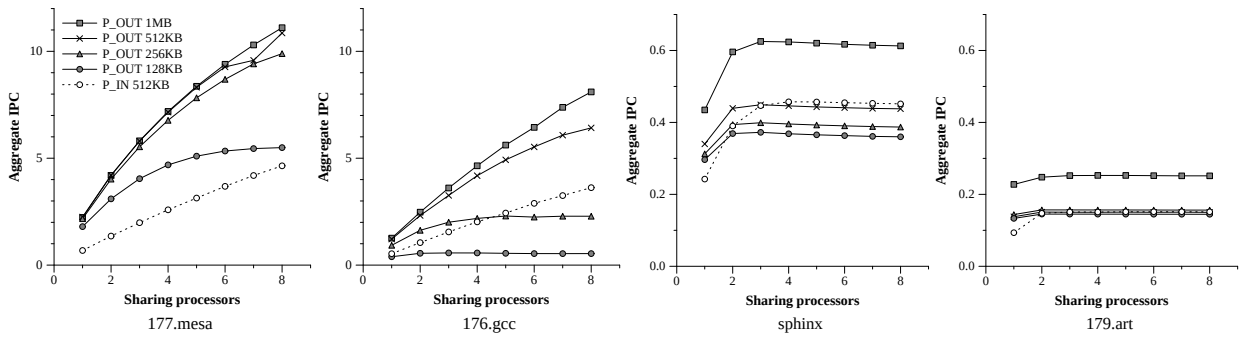


Figure 5. Performance scalability versus channel sharing.

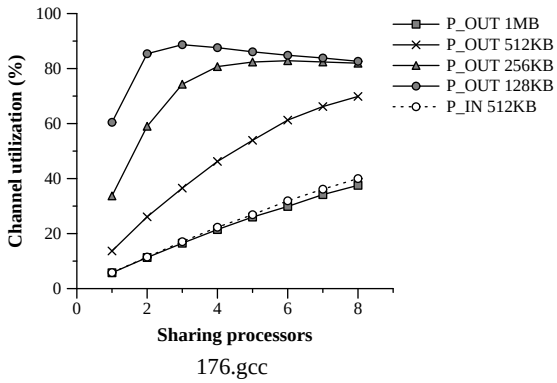


Figure 6. Channel utilization (%) versus channel sharing.

4 Maximizing CMP throughput

In the previous section, the results showed that our applications put a widely varying load on the memory subsystem, and that total job throughput levels off when the off-chip bandwidth becomes saturated. In this section, we combine our area analysis with performance simulations and our technology projections to determine which CMP configurations will be the most area-efficient for future technology generations.

4.1 Best utilization of chip area

On the left half of Table 4, we show the number of processing cores that will fit on a $400mm^2$ chip built in 70nm technology (a total of 15.5 million CBE). As the per-processor caches grow larger, the relative differences between the areas for the P_{IN} and P_{OUT} processors decline. With 128KB L2 caches, 65 P_{IN} cores and 35 P_{OUT} cores can fit on a chip, but with 1MB caches, the number of cores drops to 13 and 11, respectively. On the right half of Table 4, we show the number of processor cores that share a channel for each organization. For the 128KB, P_{IN} processors, there are over three processors sharing each channel, but for large-cache designs, the number of sharers drops to one.

In Figure 7, we show how the number of cores, number of channel sharers, and cache sizes affect area efficiency. The y-axis measures total instructions per cycle across all of the processing cores on the chip, which is equivalent to performance per unit area (since the area is held constant in all experiments). We model non-integer numbers of channel sharers by having some processors share more channels than others. We do not simulate every processor on the chip, but instead simulate just enough to compute the IPC for a subset of the processors, and then scale that result to represent chip-level throughput; our the assumption is that all processors are running the same job, albeit at skewed intervals.

The four graphs in Figure 7 show, from left to right, the total chip throughput in IPC for each of the three benchmark classes (processor bound, cache sensitive, and bandwidth bound) and the total across all benchmarks. The IPCs are computed as the harmonic means of the total chip throughput for the benchmarks in that class, at each design point. We show the means for each benchmark class, since the harmonic mean IPC across all benchmarks is heavily skewed by the low IPCs of bandwidth-bound benchmarks. On each graph, we show separate lines for P_{OUT} and P_{IN}, and also show the effects of two bandwidth capacities. The first, called *limited channels*, fixes the number of pins accord-

	No. of cores		Cores/channel	
L2 cache size	P_{IN}	P_{OUT}	P_{IN}	P_{OUT}
No L2	139	50	6.6	2.4
64KB	89	41	4.2	1.9
128KB	65	35	3.1	1.7
256KB	42	27	2.0	1.3
512KB	25	19	1.2	1
1MB	13	11	1	1

Table 4. Number of cores and cores/channel.

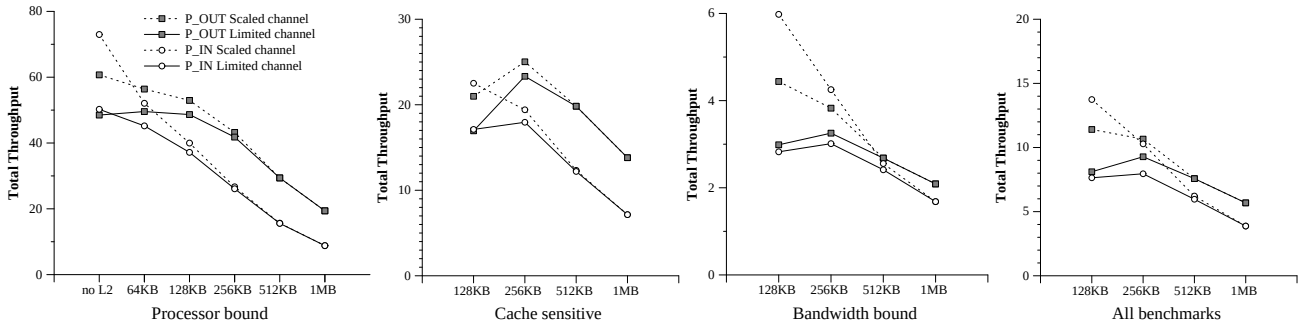


Figure 7. Best configurations.

ing to the SIA projections, and divides 4-byte wide, 60-pin channels up among the processors on the chip (channel sharing). The second model, called *scaled channels*, assumes that the processor pin counts can be scaled to provide one 60-pin channel per processing core, no matter how many cores exist on the die.

For the processor-bound benchmarks, the most area-efficient configuration uses P_{OUT} cores with 64KB L2 caches. Since those benchmarks are largely compute-bound, the additional cache provides insufficient benefit to justify the area it consumes. With scaled channels, however, the organization that achieves peak throughput is P_{IN} with no L2 cache. That organization, however, would contain 140 processing cores, requiring 8400 signal pins.

The cache-sensitive applications show a different result, with the best configuration using P_{OUT} cores with 256KB L2 caches. Enough of the working sets are contained in the L2 caches at that point to make larger caches not worth the additional area consumed. We note that at 256KB and larger L2 caches, the gap between scaled and finite memory channels is small. At 70nm, the gap between transistor counts and pins is much smaller than at 50 or 35nm, allowing the limited channel designs to have one channel per core for the larger-cache configurations.

For the bandwidth-bound applications, the configuration using P_{OUT} processors with 256KB caches is best. However, the difference between P_{OUT} and P_{IN} chip throughput is small and constant across all cache sizes, since the applications are bandwidth-bound at all of the measured cache sizes for both types of cores. We note that the scaled channel throughputs are significantly higher than the finite pin results for the smaller cache sizes, because scaled bandwidth removes the bottleneck from the bandwidth-bound applications. Finally, across all applications, we see that the P_{OUT}, 256KB L2 combination is best for finite bandwidth. However, if each processor could have its own memory channel, regardless of the number of processors, we note that the P_{IN}, 128KB L2 organization would be best.

4.2 Technology scaling

Figure 8 shows how total throughput scales for each of the benchmark classes, as technologies shrink. Each graph shows the total chip throughput obtained by the best performance/area design at each technology, for both the limited and scaled channel organizations.

At 100nm technologies, the limited and scaled bandwidth points are similar, since the best configurations generally have few enough processors to permit one channel per core, even with a finite number of pins. However, for smaller technologies, the gap between bandwidth-constrained and bandwidth-unconstrained performance grows significantly. The actual performance achieved is related to the bandwidth demands of each application class at 35nm: the processor-bound applications sustain over 100 instructions per cycle with the best configuration, the cache-sensitive benchmarks sustain over 50 IPC, but the bandwidth-constrained applications benefit minimally, improving from about 2 to 3 IPC for the entire chip. With scaled channels, the ideal configurations result in much larger numbers of processors with smaller caches, and the chip-level IPCs are significantly higher, at about 210, 100, and 18 IPC for the processor-bound, cache-sensitive, and bandwidth-bound benchmarks, respectively.

Table 5 lists the ideal configurations for each of the points in Figure 8. For limited channels, all of the configurations use out-of-order cores, with the exception of the bandwidth-bound applications at 100nm. For scaled channels, small caches are the norm, with 128KB for all classes except for cache-sensitive, which require 256KB caches at all technologies.

For constrained bandwidth, however, the required caches grow as the number of I/O pins per transistor drops. At 35nm, even the processor-bound benchmarks show an ideal configuration of 256KB (as opposed to 128KB at all other technologies,) while the cache-sensitive and bandwidth-bound benchmarks require 512KB and 1MB L2 caches, respectively. These large caches restrict the area available

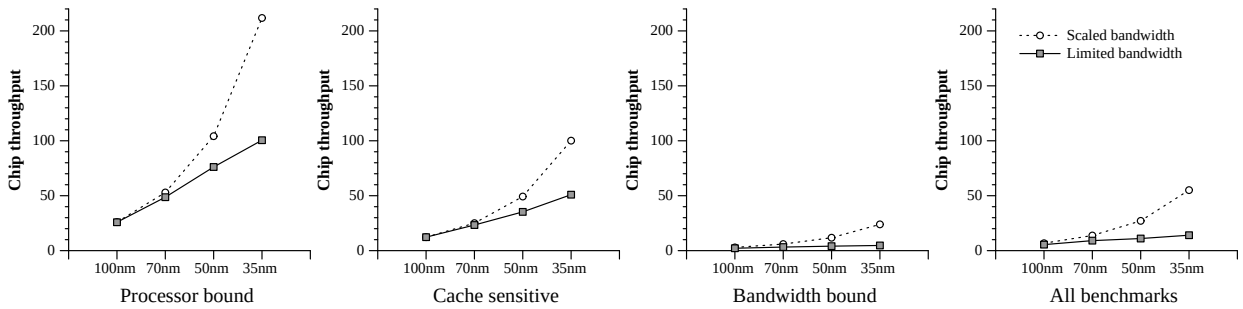


Figure 8. Total throughput versus technology.

for more processing logic, curtailing the throughput scaling severely.

For bandwidth-bound benchmarks, the best number of processors for limited channels decreases from 28 to 23 when the gate length shrinks from 70nm to 50nm. Additional area allows larger caches with a big IPC boost per core, so fewer cores with larger caches give higher total throughput.

Even for the ideal configurations, the large performance gap between limited and scaled channel organizations indicates that much of the throughput potential of future CMPs will go unrealized if solutions are not found to mitigate these bandwidth restrictions.

5 Extrapolation to server applications

This paper shows how the best allocation of resources are determined by the characteristics of applications. Given the key characteristics of any applications, including ILP and memory access patterns, we could estimate what resources, among computation power, cache, and memory bandwidth, are most critical. In this section, we briefly discuss the characteristics of database server workloads, and relate them to the applications described in section 4.

nm	Type	# cores (b/w unlim.)	Cores/ channel	Best config limited b/w	Best config b/w unlim.
100	PB	18 (18)	1.0	P_{OUT}+128KB	P_{OUT}+128KB
	CS	14 (14)	1.0	P_{OUT}+256KB	P_{OUT}+256KB
	BW	32 (32)	1.8	P_{IN}+128KB	P_{IN}+128KB
	All	18 (32)	1.0	P_{OUT}+128KB	P_{IN}+128KB
70	PB	36 (36)	1.7	P_{OUT}+128KB	P_{OUT}+128KB
	CS	28 (28)	1.3	P_{OUT}+256KB	P_{OUT}+256KB
	BW	28 (66)	1.3	P_{OUT}+256KB	P_{IN}+128KB
	All	28 (66)	1.3	P_{OUT}+256KB	P_{IN}+128KB
50	PB	71 (71)	3.0	P_{OUT}+128KB	P_{OUT}+128KB
	CS	55 (55)	2.3	P_{OUT}+256KB	P_{OUT}+256KB
	BW	23 (129)	1.0	P_{OUT}+1MB	P_{IN}+128KB
	All	23 (129)	1.0	P_{OUT}+1MB	P_{IN}+128KB
35	PB	111 (144)	4.6	P_{OUT}+256KB	P_{OUT}+128KB
	CS	77 (111)	3.2	P_{OUT}+512KB	P_{OUT}+256KB
	BW	47 (262)	1.9	P_{OUT}+1MB	P_{IN}+128KB
	All	47 (262)	1.9	P_{OUT}+1MB	P_{IN}+128KB

Table 5. Ideal configurations across technologies for finite and unconstrained bandwidth.

OLTP and DSS workloads are two of the most commonly used server workloads. As shown in the previous sections, the cache system performance, and demands on memory bandwidth, have a large impact on the balanced allocation of on-chip resources. Both of the OLTP and DSS workloads require relatively heavy loads on the memory system, compared to the computation-bound applications in our study, but OLTP and DSS have distinct characteristics [16, 19, 2, 4]. DSS workloads would easily fit into the cache-sensitive applications in our study. The L1 instruction cache miss rates of the DSS workloads are slightly higher than the average miss rates of our benchmarks, but the working sets of these workloads can fit easily in 1 or 2MB L2 caches.

OLTP workloads require a more aggressive memory system than DSS workloads do. The memory system characteristics of OLTP workloads are similar to the bandwidth-bound applications in our study. The most considerable difference is the high L1 instruction cache miss rates of OLTP workloads, as high as 20% for 8KB cache. However, the unified L2 cache performance is within the range of misses/instruction we considered in this paper. Many of the misses are shown to be conflict misses, so 8-way 2 MB L2 caches could reduce a substantial amount of the L2 misses incurred by a direct mapped 8 MB L2 cache [4].

The operating system effects on the overall performance of OLTP and DSS workloads are also different from our benchmarks. In DSS workloads, OS activity is almost negligible. In OLTP workloads, however, about 20-30% of the execution time is spent in kernel code. However, the kernel does not dominate the memory system characteristics of OLTP workloads [2]. Consequently, OLTP and DSS workloads are likely to have characteristics similar to applications presented in this paper.

Since threads in the OLTP and DSS workloads are sharing global data, our independent processes can not model

the effect of data transfers from an L2 cache bank of one core to another core. As on-chip cache sizes increase, these remote L2 cache accesses could reduce a large amount of external DRAM accesses, and have a large impact on the cache size allocation in future CMPs.

6 Related work

Recent research and development projects have begun to analyze and design chip-multiprocessor architectures. The Stanford Hydra project studied the memory hierarchy organization for a system consisting of four processing cores and on-chip caches [14]. They further proposed mechanisms for thread-level speculation, first proposed by Sohi *et al.* [23], to extract concurrency from sequential applications and to avoid sequential bottlenecks in parallel applications. Others, including Krishnan *et al.* [18] and Steffan *et al.* [24] have examined mechanisms to extract thread-level parallelism from sequential binaries for CMPs. The results from our study can be used to extend prior work in CMP architectures, contributing area and bandwidth models to the analysis.

In the server arena, the Compaq Piranha research project targets high throughput by incorporating eight light-weight single-issue in-order processors with 64KB level-1 instruction and data caches on a single chip [3]. The proposed Piranha processor has a 1MB 8-way multi-bank shared L2 cache, and each L2 cache bank has a dedicated Rambus memory channel. IBM is building actual products with the Power4 multiprocessor chip, which is optimized for commercial server workloads [10]. The Power4 consists of two out-of-order processing cores, each with a local L1 cache and sharing an on-chip level-2 cache. The access time to the on-chip L2 cache is uniform regardless of the processor on which the memory reference originates.

Additional work has examined the efficiency of memory hierarchies and proposed mechanisms to balance processor and memory system performance. Jouppi *et al.* studied the best cache size for two level cache hierarchy of single-core processors [15]. That research explored the trade-offs between miss rates and latencies of various cache sizes. Their result indicated two-level caches perform better than single-level caches with the same chip area. Farrens *et al.* studied the area efficiency of single-chip systems by comparing a single-core architecture with a large cache to a multi-core architecture with smaller caches [12]. In their results, they projected increased area efficiency for multi-core systems over single core systems with large caches.

Finally, other researchers have detailed the importance of memory bandwidth in uniprocessor architectures. Burger *et al.* studied the bandwidth scaling trends for future superscalar microarchitectures [6]. That study showed that pin bandwidth would limit performance growth rates of uniprocessors if they remained on their performance growth curve. We see similar results, except that our study uses throughput-oriented CMPs to achieve continued performance scaling. In their comparison study of different DRAM architectures, Cuppu *et al.* examined the impact of both latency and bandwidth on application performance [8]. They concluded that as processor and memory speeds continue to diverge, increasing memory bandwidth will become both critical to performance and more challenging to achieve.

7 Conclusions

Ideally, highly parallel CMP designs will offer linear scaling of throughput with increasing transistor count. In fact, job throughput may emerge as the only way to scale total chip performance for general-purpose applications, barring substantial progress and innovation that reverses the diminishing returns of current superscalar processor designs. However, limited off-chip bandwidth will always constrain the maximum number of cores that can be placed on a chip. A pressing question for CMP designers concerns the severity of limited bandwidth. In this study of the CMP design space, we have observed the following:

- Transistor counts are projected to increase considerably faster than pins, and there will be 45 times fewer pins per transistor at 35nm than at 180nm. If transistor count increases are used to increase the processor count, the number of pins per processor will decrease. Left unaddressed, that growing imbalance will drastically limit the number of cores that can be used in future technologies, and/or the throughput that can be obtained from those cores.

- Out-of-order issue cores are more area-efficient than in-order issue cores. The area ratio of P_{OUT} to P_{IN}, including 256KB L2 caches, is 1.54. Since P_{OUT} typically provides more than a 54% performance increase over P_{IN}, the out-of-order cores are more area-efficient, unless the application in question is bandwidth bound.

- For the workloads we studied, the impact of insufficient bandwidth causes the throughput-optimal L2 cache sizes to grow from 128KB at 100nm, to 256KB at 70nm, and to 1MB at 50 and 35nm. The channel contention is sufficiently severe that P_{OUT} cores with 1MB caches are more area-efficient than organizations with significantly smaller caches.

- Applications show remarkably similar behavior and performance when measured against the rate of off-chip accesses. This observation may prove useful for

estimating or modeling overall performance of a CMP on heterogeneous workloads, as a function of bandwidth demand.

The methodology of this study has some weaknesses. We are using SPEC2000 benchmarks instead of "typical" server workloads, such as web request processing or database accesses. While those workloads may have large data footprints, the results may not be qualitatively different, in terms of area efficiency, than those of the SPEC2000 benchmarks. We will measure server workloads in future work. Other refinements to this study include adjusting simulation parameters to better reflect the on-chip latencies, off-chip DRAM speeds, and processor core organizations that will likely be specific to each technology, as opposed to relying on the more conservative 70nm parameters. Finally, it is possible that power consumption will place a harder limit on chip throughput than will memory bandwidth. Since power distribution and cooling is heavily influenced by packaging technologies and cost, we did not consider heat dissipation limits in this study, but will consider it in follow-on work.

As applications become bandwidth bound, and global wire delays increase, an interesting scenario may arise. It is likely that monolithic caches cannot be grown past a certain point in 50 or 35nm technologies, since the wire delays will make them too slow. It is also likely that, given a ceiling on cache size, off-chip bandwidth will limit the number of cores. Thus, There may be useless area on the chip which cannot be used for cache or processing logic, and which performs no function other than as a placeholder for pin area. That area may be useful to use for compression engines, or intelligent controllers to manage the caches and memory channels.

Improved packaging or signaling speeds may permit greater scaling, and even larger numbers of processors, than predicted by our study. Reduced DRAM latencies would result in smaller caches, as would higher-speed pins at future technologies. If the ideal design point uses small caches, then the out-of-order cores would need a correspondingly larger performance advantage over in-order cores to remain more area efficient. A greatly improved memory subsystem might result in many more small, in-order cores being most area efficient.

In the long term, a tremendous number of processors can be designed on future CMPs to enable scaling of throughput with technology. However, setting the cache hierarchy, and number of cores *a priori* will result in poor performance across many application classes. Future CMPs would benefit from mechanisms to support adaptation to an application's available parallelism and resource needs. This *application adaptivity* is likely to be an important direction for research in future CMP designs, and is a key focus of our work.

Acknowledgments

We thank Ravi Rajwar and Jim Goodman, for providing the SMP version of the SimpleScalar code, and Weifen Lin and Steve Reinhardt for the Direct Rambus DRAM code. This work is supported by the National Science Foundation under CAREER awards CCR-9985109 and CCR-9984336, an NSF CISE Research Instrumentation grant EIA-9985991, University Partnership Awards from IBM, a Shared University Research grant from IBM, and a grant from the Intel Research Council.

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, and E. Bugnion. Memory system characterization of commercial workloads. In *The 25th Annual International Symposium on Computer Architecture*, pages 3–14, June 1998.

[3] 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.

[4] L. A. Barroso, K. Gharachorloo, A. Nowatzyk, and B. Verghese. Impact of chip-level integration on performance of OLTP workloads. In *The 6th International Symposium on High-Performance Computer Architecture*, pages 3–14, January 2000.

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

[6] 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.

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

[8] 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.

[9] W. J. Dally, M.-J. E. Lee, F.-T. An, J. Poulton, and S. Tell. High performance electrical signaling. In *The Fifth International Conference on Massively Parallel Processing Using Optical Interconnections*, June 1998.

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

[11] R. Farjad-Rad, C.-K. K. Yang, and M. Horowitz. A 0.3-um cmos 8-gb/s 4-pam serial link transceiver. *Journal of Solid-State Circuits*, pages 757–764, May 2000.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] K. Keeton, D. A. Patterson, Y. Q. He, R. C. Raphael, and W. E. Baker. Performance characterization of a quad pentium pro SMP using OLTP workloads. In *The 25th Annual International Symposium on Computer Architecture*, pages 15–26, June 1998.

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

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

[19] J. L. Lo, L. A. Barroso, S. J. Eggers, K. Gharachorloo, H. M. Levy, and S. S. Parekh. An analysis of database workload performance on simultaneous multithreaded processors. In *The 25th Annual International Symposium on Computer Architecture*, pages 35–50, June 1998.

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

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

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

[23] G. S. Sohi, S. E. Breach, and T. N. Vijaykumar. Multiscalar processors. In *The 22th Annual International Symposium on Computer Architecture*, pages 414–425, June 1995.

[24] J. Steffan, C. B. Colohan, A. Zhai, and T. C. Mowry. A scalable approach to tread-level speculation. In *The 27th Annual International Symposium on Computer Architecture*, pages 1–12, June 2000.

[25] 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.

[26] E. Yeung and M. Horowitz. A 2.4 gb/s/pin simultaneous bidirectional parallel link with per pin skew compensation. In *IEEE International Solid State Circuits Conference*, pages 256–257, February 2000.