Area and System Clock Effects on SMT/CMP Throughput

James Burns and Jean-Luc Gaudiot, *Fellow*, *IEEE*

Abstract—Two approaches to high throughput processors are Chip Multi-Processing (CMP) and Simultaneous Multi-Threading (SMT). CMP increases layout efficiency, which allows more functional units and a faster clock rate. However, CMP suffers from hardware partitioning of functional resources. SMT increases functional unit utilization by issuing instructions simultaneously from multiple threads. However, a wide-issue SMT suffers from layout and technology implementation problems. We use silicon resources as our basis for comparison and find that area and system clock have a large effect on the optimal SMT/CMP design trade. We show the area overhead of SMT on each processor and how it scales with the width of the processor pipeline and the number of SMT threads. The wide issue SMT delivers the highest single-thread performance with improved multithread throughput. However, multiple smaller cores deliver the highest throughput. Also, alternate processor configurations are explored that trade off SMT threads for other microarchitecture features. The result is a small increase to single-thread performance, but a fairly large reduction in throughput.

Index Terms—SMT, layout area estimation, processor architecture, microarchitecture trade off.

1 Introduction

Recent advances in VLSI have created challenges to continuing the current processor evolutionary trend. The smaller minimum technology spacing allows more system integration. However, most programs lack the instruction level parallelism to take advantage of the increased number of functional units. In addition, modern processors have deeper pipelines and larger relative latencies for memory accesses and routing delays. Two techniques to solve some of these problems are Chip Multi-Processing (CMP) and Simultaneous Multi-Threading (SMT). However, most current research efforts ignore the impact of silicon implementation on the performance trade off of CMP and SMT.

SMT issues instructions to functional units simultaneously from multiple threads. This creates horizontal and vertical sharing, which increases throughput through thread-level parallelism and tolerates processor and memory latencies to increase processor efficiency. The problem with a large SMT is that layout blocks and circuit delays grow faster than linear with issue width. In addition, multiple threads share the same level-1 cache, TLB, and branch predictor units, which causes contention. The resulting increase in cache misses and branch mispredict rates limits performance.

On the other hand, CMP increases layout efficiency, resulting in more functional units within the same silicon area plus faster clock rates [16]. The problem with CMP is that the hardware partition of on-chip processors restricts performance. The hardware partition results in smaller resources since the level-1 caches, TLBs, branch predictors, and functional units are divided among the multiple processors. Hence, single-threaded programs cannot use resources from the other processor cores and the smaller level-1 resources per core cause increased miss rates.

Consequently, we evaluate the Parallel On-chip Simultaneous Multithreaded processor (POSM), which is a combination of CMP and SMT. Hence, it still has the CMP advantages of more functional units and a faster clock than a wide-issue processor. The addition of SMT increases the efficiency of the underlying CMP. However, a wide-issue SMT has a microarchitectural advantage because there is no hardware partition between processor resources.

Fig. 1 shows an example of the different types of processors: superscalar, SMT, CMP, and POSM. Each box represents a potential issue to a functional unit. Processors with a faster clock have more rows of instruction, while a wider issue width has more columns for more functional units. The superscalar processor has low functional unit utilization due to data dependencies, functional unit dependencies, branch mispredictions, and cache misses. Hence, there are numerous empty issue slots where a functional unit goes unused for a cycle. SMT has the highest functional unit utilization as there are typically a number of instructions from independent threads vying to access the functional units. The CMP has the same poor utilization as the superscalar, but it has more functional units and a faster clock due to the improved layout efficiency. Merging CMP and SMT combines the advantages and disadvantages of both the individual techniques.

In this paper, we perform a detailed area estimate using routing and transistor level layout information to create four processor configurations using comparable silicon resources. We also perform a clock rate estimation using circuit simulation and delay estimates. We then run cycle accurate simulations on four processor microarchitectures

- *J. Burns is with Intel and can be reached at 19700 Alderbrook Way, Cupertino, CA 95014. E-mail: james.s.burns@intel.com.*
- *J.-L. Gaudiot is with the Department of Electrical Engineering and Computer Science, University of California, Irvine, CA 92697. E-mail: gaudiot@uci.edu.*

Manuscript received 8 Mar. 2004; revised 15 June 2004; accepted 4 Aug. 2004; published online 15 Dec. 2004.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0081-0304.

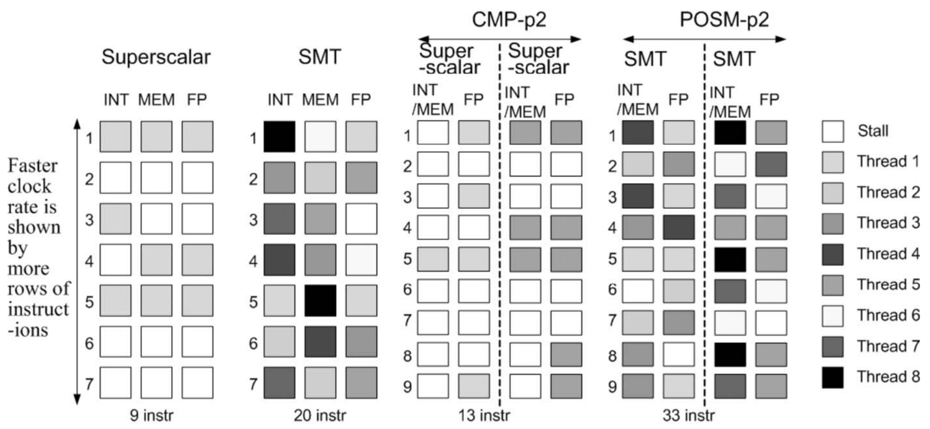


Fig. 1. CMP/SMT combinations trade microarchitecture efficiency for layout efficiency (more functional units and a faster clock, but less sharing).

to evaluate performance. The results show that area and system clock effects give an advantage to the smaller cores. In addition, the SMT overhead is larger for the wide dispatch cores because SMT increases components that grow superlinear with the processor pipeline width.

The remainder of this paper is organized as follows: Section 2 discusses related work. Section 3 shows the base CPU architecture used in this study. Section 4 describes the simulation methodology. Section 5 compares the performance of the different POSM, SMT, and CMP and CMP/SMT configurations using three different assumptions: equivalent execution resources, equivalent silicon area, and including system clock effects. Section 6 shows the resource utilization under each configuration. Section 7 gives the conclusions and summarizes the results.

2 Related Work

Prior research on SMT showed the advantages of latency tolerance and exposing parallelism through horizontal sharing [8], [13], [22], [23]. A wide-dispatch SMT was also compared to a CMP and found to have substantially higher throughput. These studies assume "equivalent execution resources" and demonstrated the advantages of horizontal sharing over the hardware-partitioned approach of CMP. The area impact from adding SMT was assumed to be negligible and was ignored in the results. The faster clock rate of a CMP was acknowledged, but was also not included in the results.

Additional research combined SMT and CMP [11]. The results showed that the combination of SMT and CMP achieved nearly the same performance as a wide-dispatch SMT, but assumed that the faster clock rate of the smaller CMP core would result in higher throughput. This paper also assumed "equivalent execution resources" and used a simple frequency scaling to adjust the clock rate. We add area effects and perform a detailed circuit analysis that results in a substantially smaller frequency penalty.

The area effect of larger processor core layout structures was shown in the Hydra [16]. A 4-core, 2-dispatch CMP was found to be equal in area to a 6-dispatch superscalar, rather than the 8-dispatch processor when using equivalent execution resources results (4-cores multiplied by 2-dispatch). The wider total dispatch width of the CMP was shown to provide higher throughput. This effort concentrated on CMP versus superscalar and did not look at SMT. System clock effects are mentioned, but are not included in the results.

Area estimates were also made on SMT processors [3]. The area overhead of SMT was shown to be substantial for eight threads, but provided a higher throughput than alternative microarchitecture techniques. The layout overhead of SMT is in addition to the overhead of the larger processor core of a wide-dispatch processor.

Detailed circuit delay and area calculations were shown in [17]. The results showed the impact of increasing relative wire delays in larger designs, as well as the impact of larger layout structures to critical paths. Our work extends the analysis to SMT.

Detailed layout area evaluations were performed on SMT processors, but not CMP [3], [5].

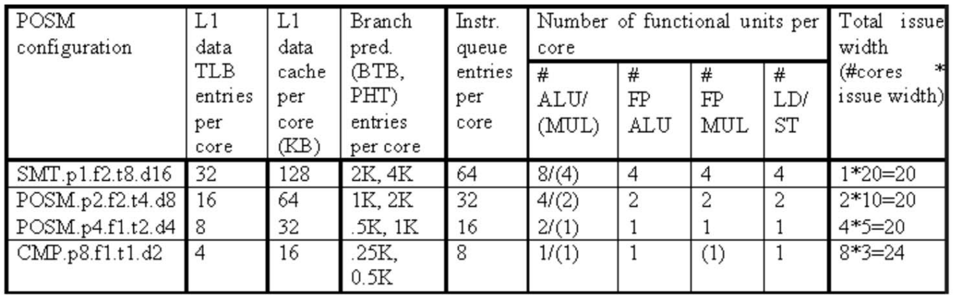
The power dissipation of SMT was also explored [20]. This compared SMT to superscalar and found the SMT had higher power efficiency. However, the power efficiency of the smaller, more efficient layout of the CMP core remains unexplored.

This research combines all of the above techniques, except for power dissipation, which is reserved for future research. If an SMT processor is compared with a CMP, then both the area and clock cycle effects of the underlying processor cores must be taken into account. Ignoring the area effects of a smaller core gives a large advantage to the wide-issue processor. Including the SMT layout overhead increases the area impact by increasing structures that grow much faster than the dispatch width. Ignoring the system clock effect also gives a large advantage to the wide-dispatch processor, while simply scaling the clock and ignoring microarchitecture enhancements gives a large advantage to the smaller CMP cores. The accuracy of the results then depends on the accuracy of the layout estimates and clock cycle analysis.

3 Processor Microarchitecture Configuration

The base processor we are using for this research is derived from the MIPS R10000 [1], [25]. We extend the R10000 by increasing its functional unit resources to create the wider

TABLE 1
Processor Configuration with Comparable Execution Units, but Unequal Silicon Area



POSM configuration	L1 data TLB entries per core	L1 data cache per core (KB)	Branch pred. (BTB, PHT) entries per core	Instr. queue entries per core	Number of functional units per core				Total issue width (#cores * issue width)
					# ALU/ (MUL)	# FP ALU	# FP MUL	# LD/ ST	
SMT.p1.f2.t8.d16	32	128	2K, 4K	64	8/(4)	4	4	4	1*20=20
POSM.p2.f2.t4.d8	16	64	1K, 2K	32	4/(2)	2	2	2	2*10=20
POSM.p4.f1.t2.d4	8	32	.5K, 1K	16	2/(1)	1	1	1	4*5=20
CMP.p8.f1.t1.d2	4	16	.25K, 0.5K	8	1/(1)	1	(1)	1	8*3=24

This gives an advantage to the wide issue processor configurations.

dispatch processors and then add SMT threads. Note that there are an unlimited number of possible configurations that could be tried. However, we maintain the processor configurations as close as possible to the prior research efforts [8], [11], [13], [22], [23]. These prior SMT/CMP papers also used a fixed pipeline. We attempt to maintain the same pipeline except where forced to add a pipeline stage due to excessive growth in critical path delay. We also maintain a maximum limit of eight SMT/CMP threads to be comparable to prior research.

The SMT processor notation is taken from [22]: SMT.1.8 corresponds to an SMT processor with a single cache-line fetch and eight threads. However, we add two additional parameters to identify the various processor configurations under study: the processor pipeline resource width and the number of on-chip processors. Hence, we have ARCH.pW.fX.tY.dZ, where ARCH is the type of microarchitecture (CMP, SMT, or POSM), pW is the number of on-chip processor cores, fX is the number of fetched cache-lines per cycle, tY is the number of threads, and dZ is the processor pipeline resource width. The pW parameter is the total number of cores per processor, while the fX, tY, and dZ parameters are per processor core.

The rest of this section details the calculations used to configure the processor for equal resources, equal area, and equal clock cycle. Both the layout area and circuit delays are based on a 0.18μm (2000 technology). The reader may skip the rest of Section 3 on the first pass through this paper and return for the detailed calculations.

3.1 Defining the Four Processor Configurations with Equivalent Execution Resources

The POSM.p4.f1.t2.d4 has four processors (p4), a single instruction fetch port (f1), two SMT threads (t2), and a four wide dispatch width (d4). The POSM.pr.f1.t2.d4 is based on a MIPS R10000 with the addition of an SMT thread. The other processor configurations are assumed to have an equivalent total set of execution resources. Hence, a processor with half the number of CMP cores has twice the functional unit resources within each core. The entire processor resources are scaled linearly, as done in prior research [8], [11], [13], [22], [23]. This approach appears to be a fair comparison since each processor configuration has an equal number of functional units. However, it ignores the fact that implementation of multiple interconnected functional units take substantially more silicon area.

There were also several changes made to the base POSM.p4.f1.t2.d4 processor to improve performance beyond the R10000. The level-2 TLB is increased by a factor of 4 because SMT places more pressure on the TLB unit. Also, a single level of larger private resources, such as a large first level TLB with no second level TLB, increases resource partitioning which would reduce performance. A smaller level-1 resource backed by a large, shared second-level resource allows more flexibility for resource sharing.

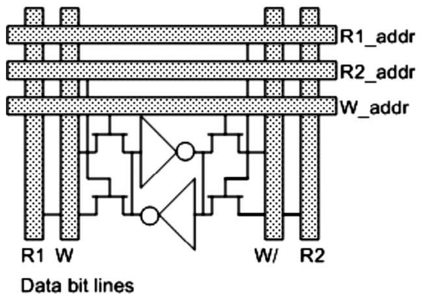
The branch predictor was originally a simple branch target buffer (BTB), but was changed to a gshare unit by adding a 1K pattern history table (PHT). The R10000-d9 has a wider issue rate and deeper pipeline, which results in a higher branch mispredict penalty. Thus, the wide issue processor makes use of the larger branch predictor (4K PHT). The original 16MB level-2 cache was too large to fit on-die in 0.18μm technology. So, an on-chip level-2 cache (512KB) is added to back up the level-1 cache and reduce the load on the external bus. An additional pipeline delay was added to the L2 for all the processor configurations with more than one core. An additional pipeline delay was added to the L2 for all the processor configurations with more than one core to allow for arbitration and routing delays. The original 16MB level-2 cache then becomes a level-3 back-side cache.

The processor cores are stepped out across the die, while only a single instance of I/O, miscellaneous, JTAG, L2 cache interface bus, MESI cache coherence, and external interface logic are needed. The CMP cache coherent interface also adds an extra pipeline stage. Table 1 shows the processor resources for a trade based on equivalent execution resources.

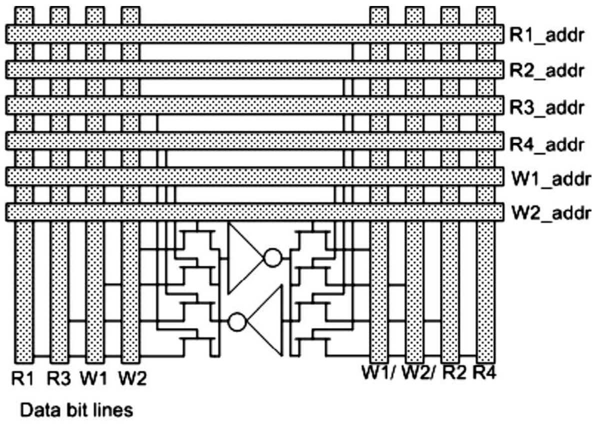
3.2 Defining Processor Configurations with Equivalent Silicon Resources

This section extends the SMT/CMP comparisons by using equivalent silicon resources as the basis. All the processor configurations are adjusted to be as close as possible in area to the POSM.p4.f1.t2.d4, but rounded to a higher processor resource width. We extrapolate the R10000 (originally designed in 1996 0.35μm technology) to 0.18μm. We then add SMT features to this design.

We calculate the SMT processor area by scaling the individual layout blocks due to the increased number of functional units and the addition of SMT threads [3]. The



(a)



(b)

Fig. 2. (a) Single-instruction issue register file with single bit read lines. (b) Dual-instruction issue register file with single bit read lines increases in both the X and Y dimensions for $O(n^2)$ area increase.

register files and remapping tables are the most critical layout blocks for SMT. The register file is a block that increases by $O(d^3)$, where d is the dispatch width, because the RAM cell grows in the X dimension due to the increased number of ports and in the Y dimension because of the added word lines and times the increased number of renaming registers. The remapping tables, register files, and instruction queue areas were calculated using a clustered layout implementation. Clustering reduces the penalty substantially for large register file arrays. However, this does add a cycle penalty when bypassing between clusters. SMT provides latency tolerance and will dispatch ready instructions from other threads, so the associated performance penalty will not be as large as in a single-threaded processor. Hence, we assumed a 0-cycle intercluster bypass penalty, giving the wide-dispatch SMT processor a slight advantage.

Our area analysis started with layout from the R10000 and from custom layouts. Multiported RAM structures were extrapolated, as shown in Fig. 2a and Fig. 2b. The RAM cell grows in the X dimension with the addition of data bit lines and in the Y dimension with additional address lines. This results in an $O(n^2)$ increase in the area of the RAM cell with increasing issue width. Alternative register file configurations are also possible to reduce area for the wider issue processor cores, but there is an associated impact to performance [19], [21].

We validated our layout estimation model by using a 256K L2 cache and comparing with the Stanford Hydra (Table 2). The starting layout size of the R10K is the same. The 6-wide dispersal processor is 180mm^2 versus 223mm^2, which is 43mm^2 or 19.3 percent more conservative. 10mm^2 of this difference is because we used 96 instruction queue entries instead of 128 and 7mm^2 because we round down when adding the FP multiplier because this is an expensive unit.

Our estimates show that the superscalar.p1.f1.t1.d8, a single core (p1), single instruction cache fetch (f1), single-thread (t1), 8-dispatch wide pipeline (d8) processor can fit within the same area that the Hydra assumed for a 6-dispatch processor. The 4-processor configurations are within 4.3 percent. We believe the clustered register file and the block-by-block layout estimate performed in this study are more accurate for the wide dispatch processors [3]. In any case, our estimates are more favorable for the wide dispersal processors, allowing more functional units within a fixed area for the larger cores.

Table 3 shows the processor core areas of the POSM configurations assuming comparable silicon resources. Column 1 gives the name of the functional block. Column 2 shows the scaled 0.18μm MIPS R10K area per functional block. The remaining columns show the SMT/CMP processor configurations. Each configuration is shown first with scalar cores and then with SMT added. The CMP.p8 already has eight multiprocessors, so SMT is not added. The subtotal for the total core processor is multiplied by the number of CMP processors in Table 4 and added to the level-2 cache, I/O, and miscellaneous overhead to arrive at the total chip area.

Table 5 gives the processor resources for the equivalent silicon resource comparison. The POSM with four processors (POSM.p4.f1.t2.d4) is the baseline. The other processor configurations are allowed to have equal or slightly larger areas. The SMT overhead increases with the core size because the larger register files and remapping tables grow at a super linear rate. The issue width increases with the number of CMP partitions because of the increased layout efficiency of the smaller core processors.

TABLE 2
Silicon Area Estimates Are Conservative Compared with Prior Research Area Estimates (Hydra), Especially for the Larger Dispath Width (d6 = Six Instruction Wide Pipeline)

Processor configuration	MIPS R10K d6 .25μm (mm^2)	MIPS R10K d6 .18μm (mm^2)	Incr. of POSM versus Hydra
Hydra.p1.f1.t1.d6	430	223	
CMP.p1.f1.t1.d6		**180**	**-19.3 %**
Hydra.p4.f1.t1.d2	424	220	
CMP.p4.f1.t1.d2		**210.5**	**-4.3%**

TABLE 3
Processor Core Area Calculation Using 0.18μm Technology

Functional Block	MIPS R10K with 512K L2 (mm^2)	Single Core Processor		Two Cores Processor		Four Cores Processor		8 Proc. Cores
		Super-scalar .p1.f1.t1.d9 (mm^2)	Add SMT {POSM .p1.f2.t8.d9 } (mm^2)	CMP .p2.f1.t1.d6 (mm^2)	Add SMT {POSM .p2.t2.t4.d6 } (mm^2)	CMP .p4.f1.t1.d4 {4 R10Ks} (mm^2)	Add SMT {POSM .p4.f1.t2.d4 } (mm^2)	CMP .p8.f1.t1.d2 (mm^2)
Dcache	5.2	26.0	26.0	13.0	13.0	5.2	5.2	2.6
Icache	5.2	20.8	26.0	10.4	13.0	5.2	5.2	2.6
TLB	1.3	10.1	12.6	4.4	5.7	1.9	1.9	0.9
Fetch, BPred	2.0	6.7	21.1	4.5	10.7	2.9	3.4	1.6
Decode	1.1	2.5	5.1	1.7	3.4	1.1	1.1	0.6
Out-of-Order execution	9.9	54.6	86.6	24.1	33.2	10.1	11.9	4.8
Register Files	2.9	12.8	40.5	5.9	12.9	2.9	4.3	1.2
Arithmetic Units	6.5	30.8	30.8	12.9	12.9	6.5	6.5	4.1
Misc.	2.4	2.4	2.4	2.4	2.4	2.4	2.4	2.4
Routing	26.4	59.3	59.3	39.5	39.5	26.4	26.4	13.2
Total Core Area (mm^2)	62.8	226.1	**310.4**	118.9	**146.8**	64.6	**68.3**	**34.8**

Scale MIPS R10K to different core sizes, then add SMT.

TABLE 4
Processor Die Size Using 0.18μm Technology

Functional Block	MIPS R10K with 512K L2 (mm^2)	Single Core Processor		Two Cores Processor		Four Cores Processor		8 Proc. Cores
		Super-scalar .p1.f1.t1.d9 (mm^2)	Add SMT {POSM .p1.f2.t8.d9 } (mm^2)	CMP .p2.f1.t1.d6 (mm^2)	Add SMT {POSM .p2.t2.t4.d6 } (mm^2)	CMP .p4.f1.t1.d4 {4 R10Ks} (mm^2)	Add SMT {POSM .p4.f1.t2.d4 } (mm^2)	CMP .p8.f1.t1.d2 (mm^2)
Core 7								34.8
Core 6								34.8
Core 5								34.8
Core 4								34.8
Core 3						64.6	68.3	34.8
Core 2						64.6	68.3	34.8
Core 1				118.9	146.8	64.6	68.3	34.8
Core 0	62.8	226.1	310.4	118.9	146.8	64.6	68.3	34.8
Misc.	6.1	8.5	6.1	6.1	6.1	6.1	6.1	6.1
Cache coherence	0	0	0	6.3	6.3	12.5	12.5	25.0
512K L2 Cache	110	110	110	110	110	110	110	110
I/O	13.7	13.7	13.7	13.7	13.7	13.7	13.7	13.7
Total CMP area	192.5	355.9	**440.1**	373.8	**429.5**	400.5	**415.4**	**432.9**

Combines multiple core areas plus interconnect, L2 cache, and I/O to create equal area CMP/SMT configurations.

Table 6 shows alternate microarchitecture configurations with nonequal layout area. The configurations in bold type are the ones used in this study. The configurations below them in italics have additional processor resources, but with a corresponding area increase. Naturally, the configurations with larger layout area would outperform the ones chosen for this study as they have more processor resources. But, this would not create an equal comparison. The SMT.p1.f2.t8.dXX configurations show the area and functional unit resources as the dispatch width is increased from d9 to d16. Increasing the dispatch width increases the resource area throughout the pipeline. If the wide-issue SMT is allowed more silicon resources, the other configurations could be increased as well. The CMP.p8.f1.t1.dX configurations could be enhanced by either increasing the dispatch width from 2 to 3 or by adding SMT.

Alternative microarchitectures for the wide-issue SMT are shown in Table 7. Here, we attempt to improve performance by trading SMT threads for other microarchitecture features. The SMT.p1.f2.t4.d11 has a single core

TABLE 5
Area Calculation Summary for Equal Processor Die Size (Comparable Silicon Resources)

POSM configuration	Area (mm^2)	SMT over-head	Instr. queue entries	Number of functional units per core				Total issue width (#cores * issue)
				# ALU/ MUL	# FP ALU	# FP MUL	# LD/ SW	
SMT.p1.f2.t8.d9	**440.1**	24%	36	5/(3)	3	2	3	1*13=13
POSM.p2.f2.t4.d6	**429.5**	15%	24	3/(2)	2	1	2	2*8=16
POSM.p4.f1.t2.d4	**415.4**	4%	16	2/(1)	1	1	1	4*5=20
CMP.p8.f1.t1.d2	**432.9**	0%	8	1/(1)	1	(1)	1	8*3=24

TABLE 6
Area Calculation Summary with Increased Functional Unit Resources and, Hence, Unequal Silicon Resources

POSM configuration	Area (mm^2)	Instr. queue entries	Number of functional units per core				Total issue width (#cores * issue)
			# ALU/ MUL	# FP ALU	# FP MUL	# LD/ SW	
SMT.p1.f2.t8.d9	**440.1**	**36**	**5/(3)**	**3**	**2**	**3**	**1*13=13**
SMT.p1.f2.t8.d10	*467*	40	5/(3)	3	2	3	1*13=13
SMT.p1.f2.t8.d12	*536*	48	6/(3)	3	3	3	1*15=15
SMT.p1.f2.t8.d16	*730*	64	8/(3)	4	4	4	1*20=20
POSM.p2.f2.t4.d6	**429.5**	**24**	**3/(2)**	**2**	**1**	**2**	**2*8=16**
POSM.p2.f2.t4.d8	*523*	32	4/(2)	2	2	2	2*10=20
POSM.p4.f1.t2.d4	**415.4**	**16**	**2/(1)**	**1**	**1**	**1**	**4*5=20**
POSM.p4.f1.t2.d5	*593*	20	3/(2)	2	1	2	4*8=32
CMP.p8.f1.t1.d2	**432.9**	**8**	**1/(1)**	**1**	**(1)**	**1**	**8*3=24**
POSM.p8.f1.t2.d2	*456*	8	1/(1)	1	(1)	1	8*3=24
CMP.p8.f1.t1.d3	*538*	12	2/(1)	1	(1)	1	8*3=24

Various processor configurations would give more performance if silicon area were ignored.

TABLE 7
Alternate Wide Issue SMT Area Calculation with Equal Layout Area Trade Off SMT Threads for Other Microarchitecture Features in an Attempt to Improve Performance

POSM configuration	Area (mm^2)	Instr. queue entries	Number of functional units per core				Total issue width (#cores * issue)
			# ALU/ MUL	# FP ALU	# FP MUL	# LD/ SW	
SMT.p1.f2.t8.d9	**440**	**36**	**5/(3)**	**3**	**2**	**3**	**1*13=13**
smt.p1.f2.t4.d11	*454.4*	*44*	*6/(3)*	*3*	*2*	*3*	*1*14=14*
smt.p1.f2.t4.64KB.d12	*452.7*	*48*	*6/(3)*	*3*	*3*	*3*	*1*15=15*
smt.p1.f1.t4.d12	*466.6*	*44*	*8/(3)*	*4*	*4*	*4*	*1*20=20*
smt.p1.mbf2.t4.BP4x.d9	*414*	36	5/(3)	3	2	3	1*13=13
smt.p1.f1.t6.BP4x.d9	*418.0*	36	5/(3)	3	2	3	*1*13=13*

(p1), a two block fetch (f2), only four threads (t4), but an increased dispatch width of 11 (d11). The wider dispatch width increases the number of functional units and resources, as well as increasing the width of the entire pipeline. However, the reduced number of SMT threads reduces the utilization of the processor resources. SMT.p1.f2.t4.64KB.d12 has smaller, faster first level caches of 64KB, less SMT threads, but an increased dispatch width of 12. SMT.p1.f1.t4.d12 has only four threads and only a single block instruction cache fetch, but maintains the full 128KB L1 cache. SMT.p1.f1.t6.BP4x.d9 has six threads (t6), a single block fetch (f1), and the baseline dispatch width of nine instructions (d9), but increases the branch prediction structures by four times (BP4x). SMT.p1.mbf2.t4.BP4x.d9 has four threads and a 4x branch predictor plus adds a single-thread two-block fetch mechanism that predicts two fetch blocks per cycle (mbf2). The improved branch prediction and improved fetch mechanism reduce mispre-diction penalties and reduces fetch bottlenecks. All of these microarchitectures are feasible alternatives within the same silicon area, trading SMT threads or cache size for better branch prediction or more execution resources. More extensive microarchitecture options are also possible, but were beyond the scope of this paper [6], [10], [15].

TABLE 8
Bypass Delays Set the Critical Path

Processor	Functional units	Wire length	Wire delay	Delta delay from baseline 4 core processor
CMP.p8.f1.t1.d2	1 ALUgen, 1 LDST , routing	3200λ + 1400λ + 2*1000λ	6600λ = 19ps	19-51-50(mux)=-82ps
POSM.p4.f1.t2.d4	2 ALUgen, 1 LDST, routing	2*3200λ + 1400λ + 3000λ	10800λ = 51ps	0
POSM.p2.f2.t4.d6	3 ALUgen, 2 LDST, routing	3*3200λ + 2*1400λ + 5*1000λ	17400λ = 133ps	133-51+50(mux) = 132ps
SMT.p1.f2.t8.d9	3 ALUgen, 2 LDST, routing	3*3200λ + 2*1400λ + 5*1000λ	17400λ = 133ps	133-51+2*50(mux) = 182ps

Hence, there are a wide number of possible configurations. The ones used in this study were selected as having the highest performance within the given layout area of the baseline configuration. Configurations with excessive layout areas would imply an unfair advantage. For example, simply increasing the number of functional units always increases performance if layout area and clock cycle effects are ignored. We also attempted to scale the entire processor resources with the number of CMP cores. This correlates with prior research efforts for a more direct comparison and attempts to avoid creating resource bottlenecks.

3.3 Defining the Pipeline Stages and Clock Cycle

The larger the processor, the more difficult it becomes to maintain a fast clock rate without adding additional pipeline stages or more exotic microarchitecture techniques that affect the instructions per cycle (IPC) [7]. The smaller core processors have shorter pipelines and/or faster clock rates. Prior SMT/CMP research assumed a fixed pipeline and then scaled the system clock based on the increased circuit and routing delays without any further microarchitecture modifications [11]. This simplistic approach is heavily biased in favor of the smaller processor cores since it ignores possible pipeline improvements. The issue stage delay increases by 300 percent when going from the baseline POSM.p4.f1.t2.d4 to the wider issue SMT.p1.f2.t8.d9 core. We attempt to solve the critical path problems as they are created through increasing the processor core resources, but with as minimal impact to the original pipeline as possible. The process cycle time was limited by the issue and bypass stages. The penalty of an added pipeline stage was shown to be less than 2 percent for single-thread performance [23].

The processor pipelines for each of the POSM configurations are shown in Table 9. The POSM.p4.f1.t2.d4 and CMP.p8.f1.t1.d2 pipelines are derived from the MIPS R10000. However, the POSM.p2.f2.t4.d6 and the SMT.p1.f2.t8.d9 require longer pipeline stages due to larger processor area and increased logic paths. Each cell within the table represents a pipeline stage with the associated delay estimate. For example, the issue and register fetch are within the same pipeline stage within the POSM.p4.f1.t4.d4, but have been broken into separate pipeline stages in the POSM.p2.f2.t4.d6.

We create two partitions within the SMT.p1.f2.t8.d9 processor pipeline because of the extensive instruction issue logic delay, but did not include the extra stage delay of intercluster forwarding. The intercluster forwarding is less of a penalty with SMT since there is no forwarding between threads. Also, there may be other techniques to reduce the area and delay of these components and we did not want to risk penalizing the large core. The wakeup logic becomes slower as the number of entries in the instruction window grows. Also, a pass through the wakeup logic is performed for each issued instruction. The critical path values shown in Table 8 are derived from published circuit delays based on the processor dispatch width [17]. However, we add the area of the bypass muxes. The larger SMT register file dramatically increases the bypass delay when the register file is placed between the functional units. Hence, we used the alternative layout that assumed the functional units are collocated below the register file. The added delay required to interact between the functional units and the register file is included in the additional pipeline stage for register fetch and retire. The bypass delay is estimated for each of the processor configurations within Table 8. An additional adjustment of 50ps is assumed for each additional level within the source mux. The bypass delay was then added to the estimated worst-case path of the ALU. The cache related delays for the Fetch and Operand fetch stages were taken from [24]. Cache sizes greater than 32KB had to be pipelined to avoid impacting the clock cycle. Hence, Table 9 shows the POSM.p4.f1.t2.d4 has a 5-stage pipeline from fetch to operand fetch and the wide-issue SMT.p1.f2.t8.d9 processor has a 9-stage pipeline. Pipelining and clustering maintained the processor clock as close as possible to the baseline processor.

We normalize the clock rates of the processors based on the POSM.p4.f1.t2.d4 (four MIPS R10000 CMP cores) processor to give a Normalized IPC (NIPC). The processor clock cycle delay is the worst case of all the pipeline stages and is listed at the far right of Table 9. The normalized clock is based on the ratio of the system clock frequencies. The number of pipeline stages can be traded for improved frequency to any or all of the processor configurations. However, the pipeline depth also reduces performance.

TABLE 9
Critical Path Delays

Processor configuration	Fetch (ns)		Decode/ Rename (ns)		Issue (ns)	Reg. Fetch (ns)	Execute plus bypass (ns)	Operand fetch (ns)		Retire (ns)		System clock or max delay (ns)
CMP.p8.f1.t1.d2	1.16		1.20		1.16		1.22	1.16		0.55		1.22
POSM.p4.f1.t2.d4	1.30		1.30		1.30		1.3	1.30		0.65		1.30
POSM.p2.f2.t4.d6	1.2	0.3	1.40		1.32	0.8	1.43	1.2	0.3	0.75		1.43
SMT.p1.f2.t8.d9	1.4	0.4	.8	.8	1.32*	0.8*	1.48*	1.4	0.4	.85		1.48

*A clustered approach was assumed to reduce the critical path delay

4 Simulation Methodology

SMT requires a detailed simulator to accurately model the effects of pipeline and memory latency tolerance. Thus, a detailed simulation must be performed that accurately models the active instructions and their data dependencies. We started from the SimpleScalar version 2.0 simulator, which already models the pipeline effects, including wrong path execution after branch mispredicts. We then added modifications to more closely model to the R10000 [2]. Next, we added SMT support. Finally, we added multiprocessing.

A multiprogram workload is used because it is easy to model and exists for workstation and server environments. A multithreaded workload has more same-type resource bottlenecks as well as issues with extracting parallelism. A multithreaded workload also has beneficial sharing within the caches while a multiprogrammed workload does not. However, it is difficult to extract substantial parallelism from some applications [12]. We explore the processing extremes, single-threaded, and multiprogrammed workloads and discuss how POSM can be combined with other techniques to increase thread parallelism. While not explored in this research, the layout area and system clock effects are expected to affect multithreaded workloads in a similar fashion. More total functional units and a faster clock rate will improve both multiprogrammed and multithreaded performance.

The application characteristics can also impact the design trade off. Tasks with large cache footprints require larger on-die caches and less functional unit resources. Kalla et al. [9] implement a POSM.p2.f1.t2.d5 with a large 1.875MB L2 cache and on-die L3 directory and memory controller. A trade off is still required between CMP/SMT within the remaining core area.

We test several SMT processor configurations using the Spec95 benchmarks. Ten benchmarks are used, five integer and five floating-point. There are a huge number of possible combinations of benchmark runs on the various processor configurations and various numbers of threads. Thus, 10 runs with a selection of spec95 benchmarks (compress, fpppp, hydro2d, ijpeg, li, m88ksim, perl, swim, turb3d, wave) are created and then stored. The selected benchmarks are then run on each configuration. Each simulation is run for (number of threads)*(100 million instructions). We then average the 10 runs to provide an average performance result as we vary the number of threads from one to the maximum thread limit of each processor configuration. Over 500 billion instructions were executed for the final simulation run used in this study.

5 Simulation Results

The simulation results section follows the same format as Section 4: simple scaling using equivalent execution resources, impact of using equivalent silicon resources, followed by normalized system clock. Prior SMT research used comparable execution resources as the comparison basis when comparing SMT and CMP [8], [11], [13], [22], [23]. However, layout area grows much faster than linear with respect to the dispatch and issue width of the processor. Thus, using comparable silicon resources affects the performance trade off [16]. Thus, we see the results with optimistic equivalent execution resource assumptions, then apply layout constraints, and then clock cycle effects.

5.1 POSM with Comparable Execution Units

This section uses the processor configurations from Table 2 to create four processor configurations with comparable execution units. We attempted to use the same approach and similar processor configurations as done in prior research [8], [11], [13], [22], [23]. However, SMT is added to the intermediate CMP cores as well as the large processor, so each chip has an equal number of threads. CMP results can be extracted from the POSM (CMP/SMT) core results by only considering the number of threads up to the hardware limit. Thus, a CMP.p4.f1.t1.d4 is equivalent to a POSM.p4.f1.t4.d4 with only four active threads (one per processor core).

Performance increases quickly with additional threads on all the configurations (Fig. 3). The large SMT processor has the highest performance over the entire range of threads because of the flexibility of vertical and horizontal sharing. However, note that the negative area and system clock effects of SMT and larger processors have not been included.

5.2 POSM Based on Equal Silicon Area

This section extends the trade off by adding the effects of layout implementation. Hence, the processor configurations are taken from Table 5. Including the inefficiencies of larger layout structures reduces the performance advantages of the processors with larger cores.

The results are shown in Fig. 4. The SMT.p1.f2.t8.d9 has the least number of functional units, but the most horizontal sharing. Hence, it has the highest single-thread performance. However, the smaller total number of functional units limits SMT throughput.

The CMP.p8.f1.t1.d2 performance increases linearly as each available thread initiates a program. This is expected

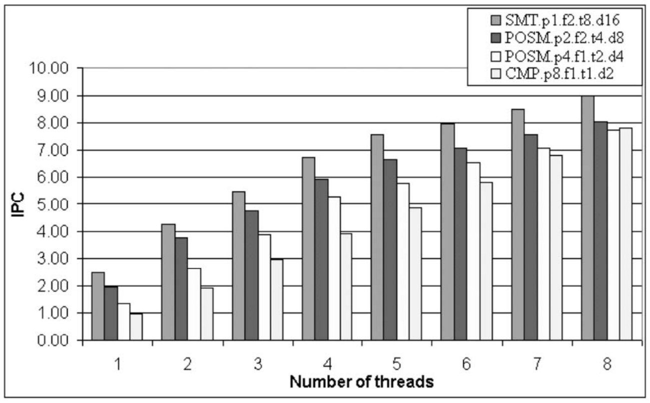


Fig. 3. Performance assuming equivalent execution resources shows microarchitectural efficiency of wide issue SMT results in best performance.

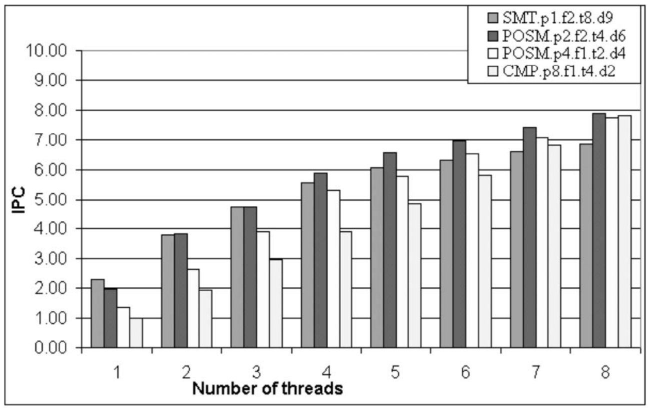


Fig. 4. Performance comparison using equivalent silicon area reduces the advantage of the wide issue processors in relation to the smaller, more layout-efficient CMP cores.

since each additional thread has a small, but private set of resources. However, it is worse than the POSM.p4.f1.t2.d4, except for eight threads, because of limited caches, branch predictors, and out-of-order logic. Therefore, the CMP.p8.f1.t1.d2 has the largest number of functional units, but they are frequently idle due to cache misses, branch mispredicts, and data dependencies. Comparing Fig. 4 with Fig. 3 shows that the layout inefficiencies of the larger processor cores reduce performance.

5.3 POSM Based on Equal Silicon Area with System Clock Effects

This section adds the circuit delay and pipeline effects discussed in Section 2.3. The results are shown in Fig. 5. Here, the IPC is reduced by the internal pipeline dependencies during simulation and then scaled by the system clock normalization factor. Thus, the SMT.p1.f2.t8.d9 and POSM.p2.f2.t4.d6 processors are negatively affected by the longer pipeline and slower system clock.

The SMT.p1.f2.t8.d9 still has the highest single-thread performance. However, the performance gap is reduced. The POSM.p2.f2.t4.d6 has 4 percent lower single-thread performance, but much higher throughput than the SMT.p1.f2.t8.d9. The CMP.p8.f1.d2 processor achieves the highest throughput.

Comparing Fig. 5 versus Fig. 4 shows that ignoring the effects of circuit delays on the processor throughput gives the larger processor cores a big advantage. This demonstrates the combined impact of adding both layout area and clock cycle affects to the CMP/SMT trade off. Greater throughput is achieved with the smaller CMP cores, although the wide issue SMT does have the best single-thread performance.

5.4 Alternate SMT Microarchitectures with Area and Clock Effect

The alternate SMT micro-architectures traded various threads for other features [3], [4], [5]. The results are shown in Fig. 6. Performance results are only possible for up to the maximum number of hardware SMT threads. Hence, the SMT.p1.f1.t4.d12 only shows results for up to four threads (t4). The additional resources also provide more opportunities for sharing between SMT threads. However, despite the increased processor resources, the lack of the additional SMT threads reduced throughput. Several microarchitecture techniques were tried: increased branch predictor resources by four times (BP4x), two block fetch from same

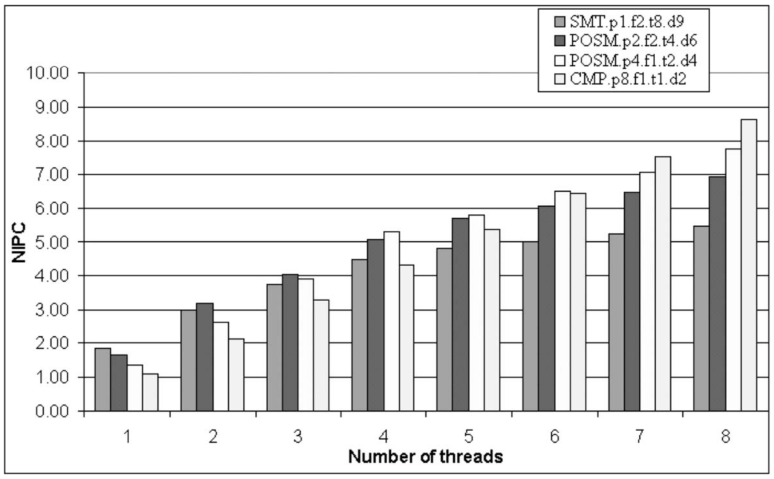


Fig. 5. Performance using equal silicon area and system clock effects shows the layout efficiency of the smaller CMP cores provides the highest throughput on a real system.

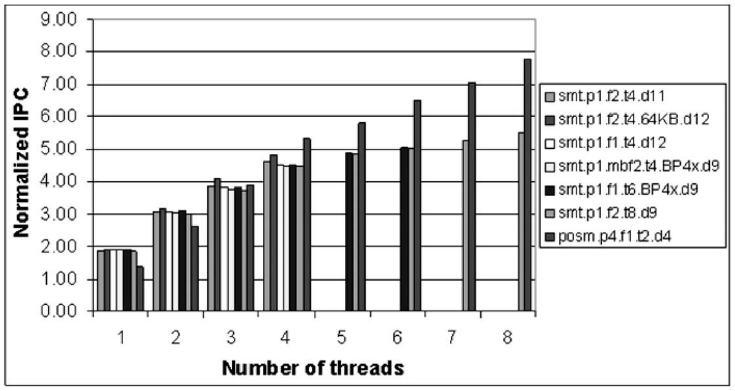


Fig. 6. Alternate SMT microarchitectures have improved single-thread performance, but reduced throughput.

thread (mbf2), and smaller first-level caches to remove a pipeline stage and provide area for other features (64KB). These changes improved single thread performance slightly. The baseline POSM.p4.f1.t2.d4 still has higher throughput and was shown as a reference.

At some point, multithreading will stop increasing throughput and will actually reduce performance due to cache and branch prediction contention, but, for SMT, it was effective for the eight threads and the SPEC95 workloads used in this study.

6 Resource Utilization

Previous studies have shown the effects of SMT on shared resources [22], [23]. We extend these results to include POSM configurations. Fig. 7 shows the miss rates for single-thread results. The hardware partition of multiple processors increases the miss rates. The level 1 data and instruction cache miss rates increase within the smaller processor cores since the cache is partitioned between the cores, resulting in a smaller per core cache. The large increase in TLB miss rates is due largely to a small number of programs whose footprint does not fit within the smaller partitioned resources.

The average single-thread results hide the variation in performance between individual benchmarks. Fig. 8 shows single-thread performance under four conditions: a best-case program with small resource demands (compress), a worst-case program with high resource demands both with and without prefetch (FPPPP), and the average of the 10 Spec95 benchmarks. The results are obtained for each processor configuration in instructions per cycle and then normalized by the processor clock rates. Performance is highest for the SMT.p1.f2.t8.d9 and drops slightly for the POSM.p2.f2.t4.d6 and begins to drop more quickly for the POSM.p4.f1.t2.d4 and the CMP.p8.f1.t1.d2.

Performance is low for the POSM.p4.f1.t2.d4 and CMP.p8.f1.t1.d2 due to the hardware partitioning. A single thread is placed on one of the POSM processors, but has no access to the resources on the other processors. Thus, a single thread on a CMP.p8.f1.t1.d2 uses only 1/8th of the total functional units. The smaller processor has a faster clock rate and a shorter pipeline than the larger processor, but this fails to compensate for such a large reduction in functional unit resources. The level-1 caches, level-1 TLB, out-of-order support, and branch prediction units are also smaller for each of the individual processors. This increases miss rates that increase memory latencies and idles functional units. However, programs with small resource demands can perform best on multiprocessors. The Compress program runs the fastest on a POSM.p4.f1.t2.d4. Compress has high cache hit rates, even on the smaller POSM.p4.f1.t2.d4 cache, and takes advantage of the faster processor clock rate.

Yet, there are some applications that perform substantially better on the larger SMT.p1.f2.t8.d9. The FPPPP benchmark has a large instruction cache footprint compared to the average Spec95 application. Hence, the instruction cache miss-rate increases dramatically as the cache size is reduced, causing poor single-threaded performance on smaller processors.

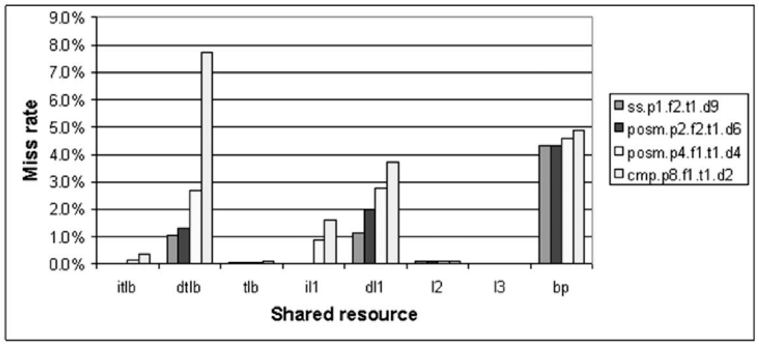


Fig. 7. Single-thread shared resource miss rates show increased miss rates due to CMP hardware partitioning.

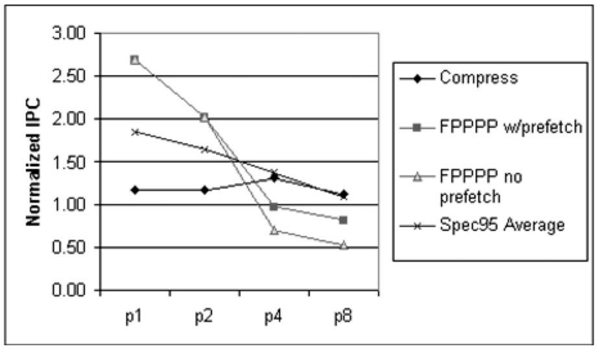


Fig. 8. Single-thread performance for small (compress), large (FPPP), and average size programs is higher on wide-issue processor cores except for some applications that fit within the resources of the smaller but faster CMP cores.

Fig. 9 shows the resource statistics for the 8-thread results. The multiple-thread statistics show the increased contention as SMT threads are added. Miss rates are fairly similar when all the processor configurations are running eight threads. The wide issue SMT has a larger cache than the smaller cores, but the larger number of threads per core increases contention between threads. However, the smaller processor cores still have higher data TLB and level 1 instruction cache miss rates because they cannot hold programs with large resource footprints. The main memory traffic has also increased in the multithreaded case, as seen by the increase in the third level cache miss rate (L3). The L3 miss rate is negligible for the CMP.p8.f1.t1.d2, but increases to 0.1 percent for the SMT.p1.f2.t8.d9 configuration. However, the L3 miss rate is still small due to the high cache hit rates of the SPEC95 application benchmarks.

7 Conclusions

Prior research has already shown that a CMP has higher throughput than a wide issue superscalar processor. Separate research has analyzed SMT and CMP processors and even CMP/SMT processors. We have extended this research by comparing more multicore processor configurations, adding layout area estimates, and including clock cycle analysis. The layout and clock cycle analysis assumptions favored the wide dispersal SMT. The SMT processor did show the best single-thread performance with improved throughput over a single-threaded processor core even when layout effects are included. However, the CMP layout efficiencies offset the microarchitecture inefficiency of fixed processor core partitions for maximum throughput performance that far surpasses the wide issue SMT processor. Relative wire delays continue to increase, making layout efficiency even more important for future fabrication technology. Our results show that:

- The SMT layout overhead increases with the processor dispatch width due to the larger register files, remapping tables, and instruction queues.
- Performance evaluations using "comparable silicon resources" give multicore processors more functional units and a higher clock rate, resulting in higher throughput than single-core SMT processors.

Our research showed the throughput potential of the POSM microarchitecture. However, substantial effort is needed in compiler research to expose more parallelism in order to reduce the runtime of a single multithreaded program. Also, the amount of parallelism was fixed at eight threads to match prior research. An additional study would be to increase the number of threads. SMT will eventually saturate the processor resources and reduce performance. The saturation point is highly dependent on the type of workload [14]. CMP layout efficiency gains are reduced as the cores become too small, plus the communication latency and routing area to the L2 cache increases. It is possible that adding SMT to the 8-core CMP would have further increased performance for the Spec95 benchmarks with minimal additional area [5].

Another area that was not discussed is power dissipation. This is a key design consideration as the number of functional units on a single die reaches the point where it is difficult to cool the device. SMT maintains a consistently high throughput, which increases the performance per watt and reduces changes in current (di/dt) while performing useful work. However, larger SMT processors also have larger layout structures. Larger layout structures have higher layout parasitics, which increases the average power dissipation. SMT also reduces wasted execution, which reduces power. Power density is also reduced through CMP, as the power is distributed across a larger portion of the die. Hot spots typically occur around power consuming units, such as the integer and floating-point units. Attempting to separate these functional units within a single, wide issue processor causes long interconnect delays. This is not an issue for CMP processors as there is already a hardware partitioning between CMP cores. Hence, the CMP hot spots are reduced compared to a wider issue core running at the

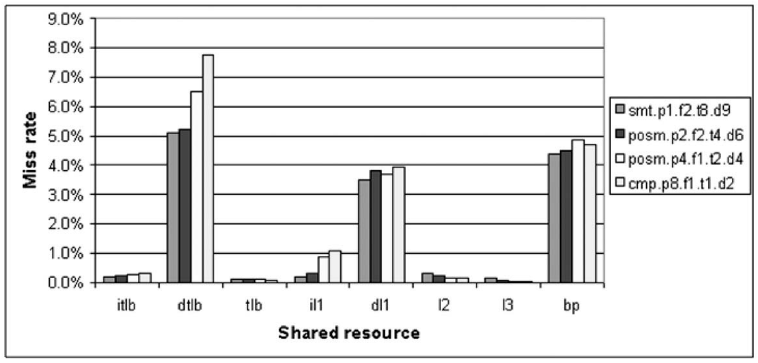


Fig. 9. Eight-thread SMT shared resource statistics show increased contention due to SMT.

same power level. This would yield an even higher throughput for a CMP core if thermal hot spots become the limiting factor. A thorough analysis is beyond the scope of this paper, but is part of ongoing research.

ACKNOWLEDGMENTS

The material reported in this paper is based upon work supported in part by the US National Science Foundation (NSF) under Grant Nos. CCR-0234444 and INT-0223647. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] A. Ahi, Y. Chen, R. Conrad, R. Martin, R. Ramchandani, M. Seddighnezhad, G. Shippen, H. Su, H. Sucar, N. Vasseghi, W. Voegtli, K. Yeager, and Y. Yeffi, "R10000 Superscalar Microprocessor," *Proc. Hot Chips Symp. VII,* Aug. 1995.
[2] D. Burger and T. Austin, "The SimpleScalar Tool Set," version 2.0, Technical Report 1342, Computer Science Dept., Univ. of Wisconsin-Madison, 1997.
[3] J. Burns and J.-L. Gaudiot, "Quantifying the SMT Layout Overhead—Does SMT Pull Its Weight?" *Proc. Sixth Ann. High Performance Computer Architecture Conf. (HPCA6),* Jan. 2000.
[4] J. Burns and J. Gaudiot, "Area and System Clock Effects on SMT/CMP Processors," *Proc. 2001 Int'l Conf. Parallel Architectures and Compilation Techniques,* 2001.
[5] J. Burns and J.-L. Gaudiot, "SMT Layout Overhead and Scalability," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 2, pp. 142-155, Feb. 2002.
[6] J. Collins, D. Tullsen, H. Wang, and J.P. Shen, "Dynamic Speculative Precomputation," *Proc. 34th Int'l Symp. Microarchitecture,* Dec. 2001.
[7] K. Farkas, P. Chow, N. Jouppi, and Z. Vranesic, "The Multicluster Architecture: Reducing Cycle Time Through Partitioning," *Proc. 30th Ann. Int'l Symp. Microarchitecture (MICRO-30),* Dec. 1997.
[8] S. Eggers, J. Emer, H. Levy, J. Lo, R. Stamm, and D. Tullsen, "Simultaneous Multithreading: A Foundation for Next-Generation Processors," *IEEE Micro,* Sept./Oct. 1997.
[9] R. Kalla, B. Sinharoy, and J.M. Tendler, "IBM Power5 Chip: A Dual-Core Multithreaded Processor," *IEEE Micro,* 2004.
[10] I. Kim and M.H. Lipasti, "Macro-op Scheduling: Relaxing Scheduling Loop Constraints," *Proc. 36th Int'l Symp. Microarchitecture,* 2003.
[11] V. Krishnan and J. Torrellas, "A Clustered Approach to Multithreaded Processors," *Proc. Int'l Parallel Processing Symp.,* Mar. 1998.
[12] M. Lam and R. Wilson, "Limits of Control Flow on Parallelism," *Proc. Int'l Symp. Computer Architecture (ISCA),* 1992.
[13] J. Lo, S. Eggers, J. Emer, H. Levy, R. Stamm, and D. Tullsen, "Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading," *ACM Trans. Computer Systems,* Aug. 1997.
[14] J. Lo, L. Barroso, S. Eggers, K. Gharachorloo, H. Levy, and S. Parekh, "An Analysis of Database Workload Performance on Simultaneous Multithreaded Processors," *Proc. 25th Ann. Int'l Symp. Computer Architecture,* 1998.
[15] O. Mutlu, J. Stark, C. Wilkerson, and Y.N. Patt, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-Order Processors," *Proc. Ninth Int'l Symp. High-Performance Computer Architecture,* 2002.
[16] K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K. Chang, "The Case for a Single-Chip Multiprocessor," *Proc. Seventh Int'l Symp. Architectural Support for Programming Languages and Operating Systems (ASPLOS VII),* 1996, http://www-hydra.standord.edu.
[17] S. Palacharla, N. Jouppi, and J.E. Smith, "Quantifying the Complexity of Superscalar Processors," Technical Report 1328, Computer Science Dept., Univ. of Wisconsin-Madison, Nov. 1996.
[18] S. Palacharla, N. Jouppi, and J.E. Smith, "Complexity-Effective Superscalar Processors," *Proc. 24th Ann. Int'l Symp. Computer Architecture,* pp. 206-218, June 1997.
[19] I. Park, M.D. Powell, and T.N. Vijaykumar, "Reducing Register Ports for Higher Speed and Lower Energy," *Proc. 35th Ann. IEEE/ACM Int'l Symp. Microarchitecture,* 2003.
[20] J. Seng, D. Tullsen, and G. Cai, "Power-Sensitive Multithreaded Architecture," *Proc. 2000 Int'l Conf. Computer Design (ICCD),* 2000.
[21] J.H. Tseng and K. Asanovic, "Banked Multiported Register Files for High-Frequency Superscalar Microprocessors," *Proc. 30th Ann. Int'l Symp. Computer Architecture,* 2003.
[22] D. Tullsen, S. Eggers, and H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Proc. 22nd Ann. Int'l Symp. Computer Architecture,* 1995.
[23] D. Tullsen, S. Eggers, J. Emer, H. Levy, J. Lo, and R. Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," *Proc. 23rd Ann. Int'l Symp. Computer Architecture,* May 1996.
[24] S. Wilton and N. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches," Western Research Laboratory Research Report 93/5, July 1994.
[25] K.C. Yeager, "The MIPS R10000 Superscalar Microprocessor," *IEEE Micro,* Apr. 1996.

James Burns received the BS degree in electrical engineering in 1982 from the University of California at Los Angeles and the MS and PhD degrees in electrical engineering from the University of Southern California in 1993 and 2000, respectively. He has worked for 19 years performing design and layout of microprocessors at TRW, Redondo Beach, California (1981-2000). He is currently working as a senior staff member designing processors for future generation IPF products at Intel, Santa Clara, California. His research interests include high-throughput processors using Simultaneous Multithreading (SMT) and on-Chip Multiprocessing (CMP), which was funded through fellowships from TRW and Intel. His recent work relates to low-power microarchitecture.

Jean-Luc Gaudiot received the *Diplôme d'Ingénieur* from the *École Supérieure d'Ingénieurs en Electrotechnique et Electronique, Paris*, France, in 1976 and the MS and PhD degrees in computer science from the University of California, Los Angeles, in 1977 and 1982, respectively. He is currently a professor and chair of the Department of Electrical Engineering and Computer Science at the University of California, Irvine (UCI). Prior to joining UCI in January 2002, he was a professor of electrical engineering at the University of Southern California since 1982, where he served as director of the Computer Engineering Division for three years. He has also done microprocessor systems design at Teledyne Controls, Santa Monica, California (1979-1980), and research in innovative architectures at the TRW Technology Research Center, El Segundo, California (1980-1982). His research interests include multithreaded architectures, fault-tolerant multiprocessors, and implementation of reconfigurable architectures. He has published more than 170 journal and conference papers. He served as editor-in-chief of the *IEEE Transactions on Computers* (1999-2002). He is a member of the ACM, of ACM SIGARCH, and a fellow of the IEEE.