Area and System Clock Effects on SMT/CMP Processors

James Burns
Intel
Santa Clara, California
james.s.burns@intel.com

Jean-Luc Gaudiot
University of Southern California
Los Angeles, California
gaudiot@usc.edu

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 multi-thread throughput. However multiple smaller cores deliver the highest throughput.

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).

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 [9]. 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 causes increased miss rates.

Consequently, we design the Parallel On-chip Simultaneous Multithreaded processor (POSM), which is an efficient 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.

Figure 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. SMT has the highest dispatch utilization. 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. The POSM combines the advantages and disadvantages of both CMP and SMT.

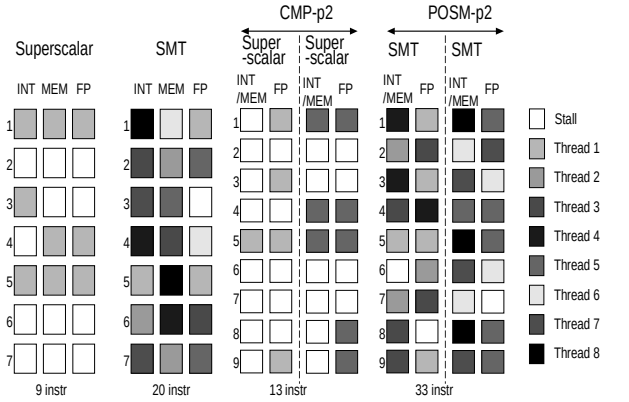

Figure 1. POSM combines CMP and SMT

In this paper, we perform a detailed area estimate using routing and transistor level layout information to create our 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 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 dispatch 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 3 different assumptions: equivalent execution resources, equivalent silicon area, and including system clock effects. Section 6 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 [5, 8, 12]. 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 [6]. 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 affects 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 [9]. 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 8 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.

The power dissipation of SMT was also explored [11]. 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 that 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 this effect. 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, 15]. We extend the R10000 by increasing its functional unit resources to

create the wider 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 [5, 6, 8, 12]. 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.

The SMT processor notation is taken from [12]: SMT.1.8 corresponds to an SMT processor with a single cache-line fetch and 8 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.

3.1. Defining the four processor configurations with equivalent execution resources

The POSM.p4.f1.t2.d4 has four R10000 processors. The other processor configurations are assumed to have an equivalent total set of execution resources. The entire processor resources are scaled linearly as done in prior research [5, 6, 8, 12].

The second level translation look-aside buffers (TLB) and branch prediction unit had additional changes prior to scaling to the different processor configurations. 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 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 is also changed from a branch target buffer (BTB) to a gshare unit with 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). An on-chip level-2 cache (512KB) is added to backup the level-1 cache and reduce the load on the external bus. The original 16MB level-2 cache becomes a level-3 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 cache coherent interface also includes 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 (2000 technology). 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 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

Table 1. Processor configuration with comparable execution units

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 (#proc * issue)
					# 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	3/(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

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.

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% 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 an 8-dispatch processor (POSM.p1.f1.t1.d8) can fit within the same area that the Hydra assumed for a 6-dispatch processor. The 4-processor configurations are within 4.3%. 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 area results of the POSM configurations assuming comparable silicon resources. Columns 1-2 give the name of the functional block and the scaled 0.18µm MIPS R10K. The remaining columns show the SMT/CMP processor configurations. Each

Table 2. 6-wide dispersal processor with 256K cache size compared with prior research

Processor configuration	MIPS R10K d6 .25µ (mm^2)	MIPS R10K d6 .18µ (mm^2)	Incr. of POSM versus Hydra
Hydra.p1.f1.t1.d6	430	223	
CMP.p1.f1.t1.d6		180	-19.3 %
CMP.p1.f1.t1.d8		218	-2.5%
Hydra.p4.f1.t1.d2	424	220	
CMP.p4.f1.t1.d2		210.5	-4.3%

Table 3. POSM configuration summary using .18•m technology

Functional Block	MIPS R10K with 512K L2 (mm^2)	Single processor		Two processor CMP		Four Processor CMP		8 Proc.
		Super-scalar .p1.f1. t1.d9 (mm^2)	POSM .p1.f2. t8.d9 with SMT (mm^2)	CMP .p2.f1 .t1.d6 (mm^2)	POSM .p2.t2 .t4.d6 with SMT (mm^2)	CMP .p4.f1 .t1.d4 {4 R10Ks} (mm^2)	POSM .p4.f1 .t2.d4 with SMT (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 Processor	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

Table 4. Area calculation summary for 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

configuration is shown first with scalar cores and then with SMT added. The CMP.p8 already has 8 multiprocessors so SMT is not added. The sub-total for the total core processor is multiplied by the number of CMP processors and added to the level-2 cache, I/O, and miscellaneous overhead to arrive at the total chip area.

Table 4 gives the processor resources for the equivalent silicon resource comparison. The POSM with 4 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.

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) [4]. 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 [6]. 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% when going to a larger processor core. We attempt to solve the critical path problems as they are created through increasing the core processor resources, but with as minimal impact to the original pipeline as possible. Our calculations including pipelining and clustering shows a system clock increase of about 13% for each doubling of the processor area. The penalty of an added pipeline stage was shown to be less than 2% for single-thread performance [13].

The processor pipelines for each of the POSM configurations are shown in Table 5. The POSM.p4.f1.t4.d4 and CMP.p8.f1.t1.d2 pipelines are taken directly 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.

We create two partitions within the SMT.p1.f2.t8.d9 processor because of the extensive instruction issue logic delay, but did not include the extra stage delay of inter cluster forwarding. 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 5 are derived from published circuit delays [10, 14].

We normalize the clock rates of the processors based on the POSM.p4.f1.t2.d4 processor to give a Normalized IPC (NIPC).

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. First, we need to model the SimpleScalar version 2.0 simulator more closely to the R10000 [2]. Second, we add SMT support. Finally, we add multi-processing.

A multi-program workload is used because it is easy to model and exists for workstation and server environments. A multi-threaded workload has more same-type resource bottlenecks as well as issues with extracting parallelism. A multi-threaded workload also has beneficial sharing within the caches while a multi-programmed workload does not. We explore the processing extremes, single-threaded and multi-programmed workloads and discuss how POSM can be combined with other techniques to increase thread

Table 5. Critical path delays

Processor configuration	Fetch (ns)	Decode (ns)	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	1.15	.6	.55	.39	.58	1.2	1.16	.55	1.2
POSM.p4.f1.t2.d4	1.3	.65	.65	.65	.65	1.3	1.3	.65	1.3
POSM.p2.f2.t4.d6	1.46	.75	.7	1.32	.8	1.4	1.6/2**	.75	1.46
SMT.p1.f2.t8.d9	1.63	.80	.8/2*	2.63/2*	1.0/2*	1.6	1.9/2**	.85	1.63

*The divide by 2 is because the delay for this SMT processor stage assumes a clustered layout structure.
**These operand fetch accesses were pipelined and have two cycles to complete.

parallelism. While not explored in this research, the layout area and system clock effects are expected to affect multi-threaded workloads in a similar fashion. More total functional units and a faster clock rate will improve both multiprogrammed and multithreaded performance.

We test several SMT processor configurations using the Spec95 benchmarks. 10 benchmarks are used, 5 integer and 5 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). A total of 360 billion instructions are run for this study.

5. Simulation results

Prior SMT research used comparable execution resources as the comparison basis when comparing SMT and CMP [5,6,8,12]. 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 [9].

This section follows the same format as section 4: simple scaling using equivalent execution resources, impact of using equivalent silicon resources, and normalized system clock. Thus we see the results with optimistic 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 and is similar to prior research [5, 6, 8, 12]. However, SMT is added to the intermediate CMP cores as well as the large processor, so each chip has an equal number of threads [12]. CMP results can be extracted from the POSM core results by only considering the number of threads up to the hardware limit. Thus a CMP.p2.f1.t1.d2 is equivalent to a POSM.p2.f2.t4.d4 with only two threads (one per processor core).

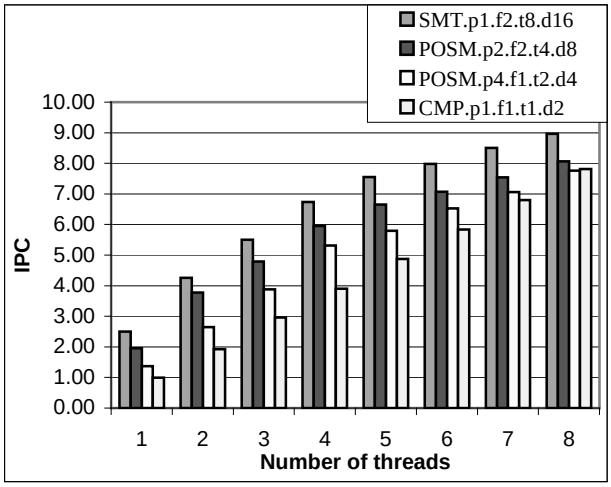

Figure 6. Performance assuming equivalent execution resources.

Performance increases quickly with additional threads on all the configurations (figure 6). 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 tradeoff by adding the effects of layout implementation. Hence the processor configurations are taken from Table 4. Including the inefficiencies of larger layout structures reduces the performance advantages of the processors with larger cores.

The results are shown in figure 7. 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 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 8 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 figure 7 with figure 6 shows that the layout inefficiencies of the larger processor cores reduce performance.

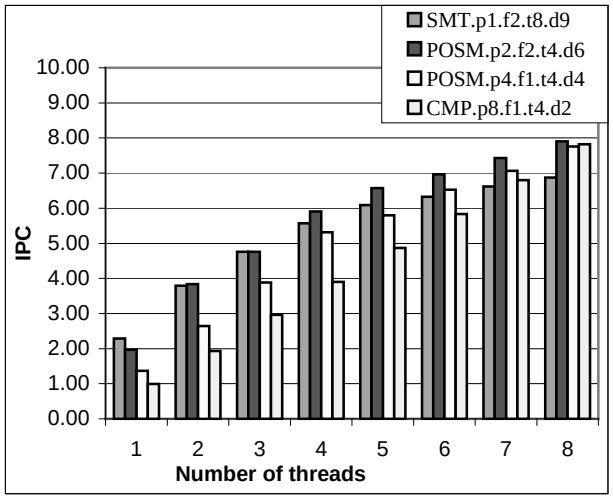

Figure 7. Equal silicon area performance comparison

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 figure 8. 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% 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.

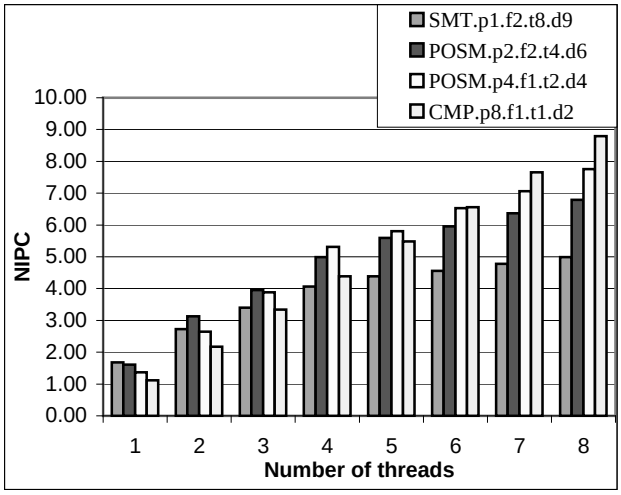

Figure 8. Equal silicon area and system clock effects

Comparing figure 8 versus figure 7 shows that ignoring the effects of circuit delays on the processor throughput gives the larger processor cores a big advantage.

6. Conclusion

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 multi-core processor configurations, adding layout area estimates, and including clock cycle analysis. The layout and clock cycle analysis 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.

In more detail, our results show that:

- Adding pipeline stages resulted in a 13% clock rate reduction when doubling the processor core, rather than 170% from simply scaling based on the critical path.
- 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” gives multi-core 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 run time of a single multi-threaded program.

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, the larger SMT processor also has larger layout structures. Larger layout structures have higher layout parasitics, which increases the average power dissipation. SMT also reduces wasted execution, which reduces power. A thorough analysis is beyond the scope of this paper, but is part of ongoing research.

7. Acknowledgements

The material reported in this paper is based upon work supported in part by the National Science Foundation under Grants No. CSA-0073527 and INT-9815742. 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 National Science Foundation.

Bibliography

[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, Yeffi, "R10000 Superscalar Microprocessor," Hot Chips Symposium VII, Stanford, California, August 1995.

[2] T. Austin, "SimpleScalar Architectural Research Tool Set, Version 2.0." http://www.cs.wisc.edu/~mscalar/simplescalar.html.

[3] J. Burns, J-L Gaudiot, "Quantifying the SMT Layout Overhead – Does SMT Pull Its Weight?" Proceedings of the 6th Annual High Performance Computer Architecture (HPCA6), January 2000.

[4] K. Farkas, P.Chow, N.Jouppi, Z. Vranesic, "The Multicluster Architecture: Reducing Cycle Time Through Partitioning." Proceedings of the 30th Annual International Symposium on Microarchitecture. MICRO-30. December 1997.

[5] S. Eggers, J. Emer, H. Levy, J. Lo, R. Stamm, D. Tullsen, "Simultaneous Multithreading: A Foundation for Next-generation Processors", IEEE Micro, September/October 1997.

[6] V. Krishnan, J. Torrellas, "A Clustered Approach to Multithreaded Processors." International Parallel Processing Symposium, March, 1998.

[7] M. Lam, R. Wilson, "Limits of Control Flow on Parallelism." ISCA 1992.

[8] J. Lo, S. Eggers, Joel Emer, Henry Levy, Rebecca Stamm, Dean Tullsen, "Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading." ACM Transactions on Computer Systems, August 1997.

[9] K. Olukotun, B. Nayfeh, L. Hammond, Ken Wilson, and Kunyung Chang, "The Case for a Single-Chip Multiprocessor." Proceedings Seventh International Symposium of Architectural Support for Programming Languages and Operating Systems (ASPLOS VII) 1996, http://www-hydra.standord.edu.

[10] S. Palacharla, Norman Jouppi, J. E. Smith, "Quantifying the complexity of superscalar processors," Tech. Rep. 1328, University of Wisconsin-Madison, CS Department, November 1996.

[11] J. Seng, D. Tullsen, G. Cai, "Power-Sensitive Multithreaded Architecture." Proceedings of the 2000 International Conference on Computer Design, ICCD 2000.

[12] D. Tullsen, Susan Eggers, Henry Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism." Proceedings of the 22nd Annual International Symposium on Computer Architecture, 1995.

[13] D. Tullsen, Susan Eggers, Joel Emer, Henry Levy, Jack Lo, Rebecca Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor." Proceedings of the 23rd Annual International Symposium of Computer Architecture, May, 1996.

[14] S. Wilton and N. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches." Western Research Laboratory Research Report 93/5.

[15] K.C. Yeager, "The MIPS R10000 superscalar microprocessor", IEEE Micro, April 1996.