Efficiency of Thread-Level Speculation in SMT and CMP Architectures - Performance, Power and Thermal Perspective

Venkatesan Packirisamy, Yangchun Luo, Wei-Lung Hung, Antonia Zhai, Pen-Chung Yew and Tin-Fook Ngai†
University of Minnesota, Minneapolis. †*Intel Corporation*
{*packve,yluo,whung,zhai,yew*}*@cs.umn.edu* *tin-fook.ngai@intel.com*

***Abstract*— Computer industry has adopted multi-threaded and multi-core architectures as the clock rate increase stalled in early 2000's. However, because of the lack of compilers and other related software technologies, most of the general-purpose applications today still cannot take advantage of such architectures to improve their performance. Thread-level speculation (TLS) has been proposed as a way of using these multi-threaded architectures to parallelize general-purpose applications. Both simultaneous multithreading (SMT) and chip multiprocessors (CMP) have been extended to implement TLS. While the characteristics of SMT and CMP have been widely studied under multi-programmed and parallel workloads, their behavior under TLS workload is not well understood. The TLS workload due to speculative nature of the threads which could potentially be rollbacked and due to variable degree of parallelism available in applications, exhibits unique characteristics which makes it different from other workloads. In this paper, we present a detailed study of the performance, power consumption and thermal effect of these multithreaded architectures against that of a Superscalar with equal chip area. A wide spectrum of design choices and tradeoffs are also studied using commonly used simulation techniques. We show that the SMT based TLS architecture performs about 21% better than the best CMP based configuration while it suffers about 16% power overhead. In terms of Energy-Delay-Squared product (ED^2), SMT based TLS performs about 26% better than the best CMP based TLS configuration and 11% better than the superscalar architecture. But the SMT based TLS configuration, causes more thermal stress than the CMP based TLS architectures.**

I. Introduction

Continuous clock rate improvement on microprocessors in the past three decades has stalled in early 2000's because of power and thermal considerations. It prompted computer industry to adopt multi-threaded (e.g. simultaneous multi-threading (SMT), hyper-threading) , and/or multi-core (e.g. chip multiprocessors (CMP)) architectures in the hope of continuing the performance improvement without increasing the clock rate and its associated power and thermal problems. However, because of the lack of compilers and other related software technologies, most of the general-purpose applications today still cannot take advantage of such architectures to improve their single-application performance.

Hardware support for speculative threads has been proposed to take advantage of multi-threaded architectures. One of the main thrusts for such an approach is to improve the performance of a single applications through *thread-level speculation* (TLS) [1]. However, there is a significant lack of a understanding on how various multi-threaded architectures and their implementations interact with TLS on general-purpose benchmarks.

Both CMP and SMT processors have been extended to support TLS. In the case of CMP, one popular approach is to buffer speculative stores in the *local* L1 cache, and extend the existing cache coherence protocols to detect data dependence violations [1] (we refer to this architecture as the *CMP-TLS architecture*). In the case of SMT, the *shared* L1 cache is augmented with extra bits for the same tasks [2]. We refer to this architecture as the *SMT-TLS architecture*. Even though there have been numerous studies on the performance aspects of CMP-TLS and SMT-TLS architectures, there has not been a detailed comparative study on their *performance, power and thermal effects* when compared to Superscalar architecture *under the constraint of same chip area*. Such detailed study is essential to identify the issues in the different multithreaded architectures which in turn would help in efficient TLS architecture design.

CMP and SMT architectures have been studied in detail under multi-programmed and parallel workloads [3], [4], [5], [6], but the same conclusions are not applicable for TLS workloads due to its unique characteristics. For example, in SMT-TLS, speculative and non-speculative threads share the same core which could lead to better resource utilization. But the speculative threads could also slow down the non-speculative threads by competing for resources with the non-speculative thread. The shared cache in SMT-TLS allows all threads to share the same working set as they are working on the same single application. This could lead to a better cache performance due to prefetching. But also, as speculative state from all speculative threads are buffered in the shared cache in SMT-TLS, it is more susceptible to stalls by conflict misses as the cache lines holding the speculative states cannot be evicted. Also, speculative threads could be preempted to free cache lines for older speculative threads leading to an increase in the number of squashes in SMT-TLS.

Given the unique characteristics of TLS workload, it is impossible for us to infer whether the TLS workload is more efficient on CMP or SMT processors in terms of performance, power and thermal effect when same chip area is used. This paper, to the best of our knowledge, presents the first thorough study of performance, power, energy-delay-product and thermal effects of a general-purpose workload, generated by a TLS parallelizing compiler, on SMT and

CMP architectures under equal die-area constraint. A wide spectrum of design choices and tradeoffs are studied using commonly used simulation techniques.

Our results show that the main drawback for CMP is its poor performance in code regions with low thread-level parallelism, while the main drawback for SMT is its core complexity and frequent squashes due to buffer overflow leading to higher power consumption. Different applications, depending on their specific characteristics prefer different architectures. Out of the 15 SPEC 2000 benchmarks considered, 5 benchmarks prefer CMP architecture while the remaining benefit from SMT architecture leading to about 26% better ED^2 for SMT over the CMP based TLS architecture. In terms of thermal behavior, across all benchmarks CMP architecture shows lower thermal stress than the SMT architecture.

The rest of the paper is organized as follows: Section II describes the related work. Section III considers various trade-offs and configures the three architectures, Superscalar, SMT and CMP, with equal die area; Section IV describes our evaluation methodology; Section V evaluates the performance and energy-delay-product of each architecture under TLS workload; Section VI studies the sensitivity of these results with several key architectural parameters; Section VII presents the thermal effects of the TLS-workload on the three architectures; and in Section VIII we present our conclusions.

II. Related work

While the discussions on TLS performance have mostly been under the context of CMP [7], [8], [9], SMT processors can also be extended to support TLS [10], [2]. However, given the characteristics of TLS workload described earlier, it is not clear which architecture can achieve a higher performance and a better power efficiency while creating less thermal stress.

Renau *et. al* [11] compared the power efficiency of a CMP processor with TLS support against an equal-area, wide-issue Superscalar processor. They concluded that the CMP processor with TLS support can be more power efficient on general-purpose applications. Their selection of equal-area configurations is based on a rough assumption that a 6-issue Superscalar has the same area as a 4-core 3-issue CMP. In this paper we conduct a detailed study of area overhead to identify equal area configurations. Also we include SMT based TLS in our comparison. Warg *et. al* [12], compared speedup of SMT and CMP using simple assumptions to choose the configurations. In this paper, we study several equal area configurations based on detailed area estimation. Also we present a detailed comparison which includes performance, power and thermal effects.

Numerous studies have compared the SMT and CMP performance and power efficiency under different workloads. On parallel programs [13] and mobile workloads [3], SMT processors outperform CMP processors. However, on multimedia workloads, CMP is more efficient [4]. In the context of multi-program workload, Li *et. al* [5] found that SMT is more efficient for memory-bound applications while CMP is more efficient for CPU-bound applications; Burns et al [6] found that SMT can achieve a better single thread performance, but CMP can achieve a higher throughput.

TABLE II

Die area estimation for (1) Superscalar (SEQ), (2) SMT processor with reduced complexity occupying an equal area and (3) CMP processor with an equal area as SEQ.

Hardware structures	SEQ	SMT-4	CMP-4-2MB
	area (mm^2)	area (mm^2)	area (mm^2)
Function units			
Integer units	1.296	1.134	0.648
Floating point units	1.760	1.408	0.704
Load Store units	0.551	0.551	0.367
	3.607	3.093	1.719
Pipeline logic			
Fetch unit	0.477	0.597	0.239
Decode unit	0.441	0.485	0.220
Issue unit	0.392	0.431	0.196
Writeback unit	0.392	0.377	0.196
Commit unit	0.216	0.248	0.108
Caches			
TLBs	0.129	0.142	0.104
L1 I-cache	1.748	2.397	0.439
L1 D-cache	2.519	3.808	0.569
Register file	1.361	5.057	0.414
RUU	18.325	12.134	1.925
LSQ	1.771	0.974	0.185
Misc	1.216	2.866	0.3422
Core Size	32.6	32.6	6.6
Bus area			5.95
L2 cache	50.71	50.71	50.71
Chip size	83.3	83.3	83.3

III. Processor Configurations

For fair power and performance comparisons among Superscalar, CMP-TLS and SMT-TLS architectures, we maintain the same chip area for the three different processor configurations. We use a detailed area estimation tool presented in [14]. While the original tool only targets SimpleScalar-based architectures, we have extended this tool to estimate area of SMT and CMP architectures.

However, even for a fixed chip area, many processor configurations are possible by varying the size of the cores and the caches; and it is not possible to exhaustively evaluate the entire design space. In this section, we describe how equal-area processor configurations are selected for fair comparisons in this study.

A. Superscalar configuration

Our base configuration is a SimpleScalar-based Superscalar architecture. The architectural parameter of this processor can be found in Table I. The die area occupied by each component of this processor can be found in Table II, estimated by the die-area estimation tool [14] (assuming 70nm technology). We refer to this architecture as the *SEQ architecture*, since it executes sequential programs.

B. SMT configuration

The SMT architecture is based on the Simultaneous Multithreading architecture proposed by Lo *et. al* [13], where processor resources are fully shared by all threads. Up-to two threads are allowed to fetch instructions in the same cycle

TABLE I

ARCHITECTURAL PARAMETERS FOR THE SUPERSCALAR (SEQ) CONFIGURATION AND THE SMT CONFIGURATIONS WITH 2 AND 4 THREADS

Parameter	Superscalar	SMT4	CMP-4-2MB
Fetch/Decode/Issue/Retire Width	12/12/8/8	12/12/8/8	6/6/4/4
Integer units	8 units / 1 cycle latency	7 units	4 units
Floating point units	5 units / 12 cycle latency	4 units	2 units
Memory ports	2Read, 1Write ports	2R,1W	1R and 1W
Register Update Unit (ROB,issue queue)	256 entries	185	105
LSQ size	128 entries	80	42
L1I Cache	64K, 4 way 32B	64K, 4 way 32B	16K, 4 way 32B
L1D Cache	64K, 4 way 32B	64K, 4 way 32B	16K, 4 way 32B
Cache Latency	L1 1 cycle, L2 18 cycles		
Unified L2	2MB, 8 way associative, 64B blocksize		
Physical registers per thread	128 Integer, 128 Floating point and 64 predicate registers		
Thread overhead	5 cycles fork, 5 cycles commit and 1 cycle inter-thread communication		

based on the *icount* fetch policy. Hardware support of TLS is implemented by extending the shared L1 cache to buffer speculative states and track inter-thread data dependences [2].

The overall area cost for supporting a four thread SMT processor (SMT-4) with the same configuration as Superscalar (SEQ) is approximately 30% (estimated based on our tool). To configure a SMT core with the same area as the SEQ configuration, we need to compensate for this overhead by reducing the complexity of the SMT core.

The complexity of the core can be reduced by reducing many parameters, but our main target is the RUU(Register Update Unit) since it occupies a significant die area (about 56% of SEQ). However, if we simply reduce the number of RUU and LSQ (Load Store Queue) entries while holding other parameters constant, we must reduce the number of RUU entries by 60%. This approach clearly creates a performance bottleneck, and thus produces a sub-optimal design. RUU requires many ports, since it is the central structure accessed by almost all pipeline stages. By reducing the number of function units, we can reduce the number ports in RUU, in turn, reduce the area cost of RUU.

In this paper, we reduce both the number of function units and the number of RUU and LSQ entries to achieve the desired area cost. The exact configuration chosen for SMT configuration is shown in Table I. In Table II, the area of each component in this equal area SMT configuration is shown.

To study the impact of the reduction in the number of TLS threads, we include a configuration called SMT-2 which supports 2 threads (equal area as SEQ and SMT-4).

C. CMP configurations

In choosing the area-equivalent CMP configurations we have two design choices. One way is to hold the L2 size the same as in SEQ and allocate less area for each core, so the total area for the multiple cores is the same as that of the Superscalar core (as in [6]). Another choice is to reduce L2 cache size and use the area for allocating more area for each core (as in [5]). Also, we could reduce the number of cores supported, which will allow us to use larger cores. To cover all these design choices, we consider four different configurations of CMP architecture - CMP-4-2MB(CMP-4cores-2MB L2 cache), CMP-4-1MB, CMP-2-2MB, CMP-2-1MB.

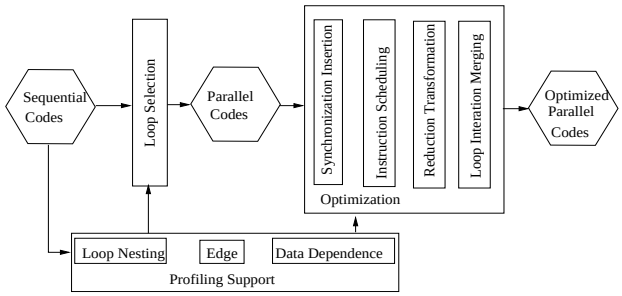


Fig. 1. Compilation infrastructure

We estimated the area of each configuration and made sure they have the same area. Due to lack of space we show only one configuration (CMP-4-2MB) in Table II. The simulation parameters for the CMP-4-2MB are shown in I.

IV. EVALUATION METHODOLOGY

We use a trace-driven, out-of-order Superscalar processor simulation infrastructure. The trace-generation portion of this infrastructure is based on the PIN instrumentation tool [15], and the architectural simulation portion is built on SimpleScalar. We not only model the register renaming, the reorder buffer, branch prediction, instruction fetching, branching penalties and the memory hierarchy performance, but also extend the infrastructure to model different aspects of TLS execution including explicit synchronization through signal/wait, cost of thread commit/squash, etc.

To estimate power consumption of the processors, the simulator is integrated with the Wattch [16] power model. The power consumption for the common bus in the CMP architectures is simulated using Orion [17]. The power traces generated by the simulator are fed to HotSpot [18] to evaluate the thermal behavior of the system.

We evaluated all SPEC2000 benchmarks written in C.(except GAP). Statistical information on the set of loops selected for each benchmark can be found in Table III.

A. Compilation Infrastructure

Our compiler infrastructure [19] is built on Intel's Open Research Compiler(ORC) [20], an industrial-strength open-source compiler targeting Intel's Itanium Processor Family (IPF). To create efficient speculative parallel threads, compiler must perform accurate performance trade-off analysis to determine whether the benefit of speculative parallel execution outweighs the cost of failed speculation. In our

TABLE III
DETAILS OF BENCHMARKS

Benchmark	No of loops selected	coverage of selected regions	No of samples
perlbmk	9	23%	13
art	25	99%	12
vpr_place	3	53%	12
gcc	98	80%	21
parser	40	37%	18
vpr_route	19	89%	14
mcf	13	98%	10
equake	9	91%	21
ammp	21	99%	16
twolf	20	48%	19
bzip2	19	81%	18
mesa	3	63%	15
gzip	6	99%	20
crafty	3	14%	17
vortex	8	67%	22

case, the compiler performs such analysis based on loop nesting [9], edge, as well as data dependence profiling [21], as shown in Figure 1. The parallel compiler has two distinct phases - loop selection and code optimization:

Loop Selection: In the loop selection phase, the compiler estimates the parallel performance of each loop. The compiler then chooses to parallelize a set of loops that maximize the overall program performance based on such estimations [9], [19].

Code Optimization: The selected parallel loops are optimized with various compiler optimization techniques to enhance TLS performance: (i) all register-resident values and memory-resident values that cause inter-thread data dependences with more than 20% probability are synchronized [8]; (ii) instructions are scheduled to reduce the critical forwarding path introduced by the synchronization [7], [19]; (iii) computation and usage of reduction-like variables are transformed to avoid speculation failure [19]; and (iv) consecutive loop iterations are merged to balance the workload of neighboring threads [19].

B. SimPoint Sampling

Prior TLS research typically simulated the first billion instructions in each benchmark after skipping the initialization portion. The truncated simulation does not cover all phases in a benchmark, and thus can potentially miss important program behavior that only appear in the later parts of the execution To improve simulation accuracy and to reduce simulation time, we have adopted a SimPoint-based sampling technique [22].

When running SimPoint, we use -maxK (maximum number of samples) as 30 and the sample size as 30 million instructions. The number of phases selected by SimPoint for each benchmark is shown in Table III.

V. PERFORMANCE AND POWER COMPARISONS

We compare the three different architectures - CMP-based TLS, SMT-based TLS and Superscalar in terms of performance in Section V-A. In Section V-B, we compare their power consumption, and in Section V-C, we use *energy-delay product* (ED) and *energy-delay-squared product* (ED^2) to compare energy efficiency.

A. Performance

Fig. 2(a) shows the speedup of the entire benchmark suite using Superscalar (SEQ) performance as the base and Fig. 2(b) shows the breakdown of execution time when executing loops selected by the compiler. In this section, we only show the TLS configurations: CMP-4-2MB and SMT-4. We will discuss other possible configurations in Section VI.

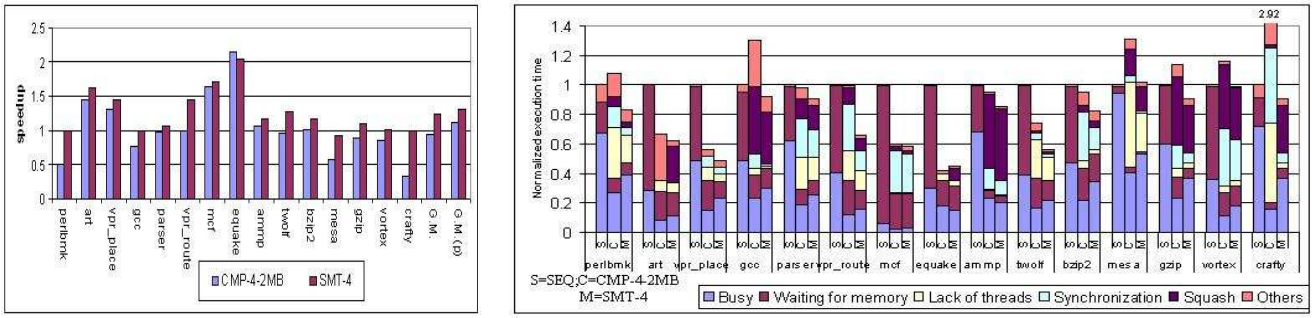
The CMP-4-2MB slows down in *perlbmk*, *gcc*, *parser*, *twolf*, *mesa*, *gzip*, *vortex* and *crafty*, leading to a *geometric mean* (GM) slowdown of 6% when compared to SEQ. But if we eliminate top three worst performing benchmarks *mesa*, *perlbmk* and *crafty*, the CMP-4-2MB achieves 12% speedup over SEQ(indicated by GM(p)). Due to its dynamic sharing of resources, SMT-4 is able to extract good performance even in benchmarks with limited parallelism except in *gcc*, *mesa* and *perlbmk*, leading to about 23% speedup over SEQ.

Each benchmark benefits from specific architecture depending on its characteristics. A comparison of the impact of different benchmark characteristics on the TLS performance in CMP and SMT architectures is presented in Table IV.

Large sequential non-parallelized code regions: The CMP-4-2MB slows down about 6% compared to SEQ but it achieved about 6% speedup if we consider only the parallel regions (in Fig. 2(b)). Many of the benchmarks considered have significant sequential (non-parallelized) regions which suffer poor performance on CMP-4-2MB due to its static partitioning of resources. The *perlbmk* shows more than 50% slowdown for CMP-4-2MB configuration. The coverage of sequential regions in *perlbmk* is about 77%. Due to this very low parallel-region coverage, we see a huge decrease in overall performance for *perlbmk*. In benchmark *twolf*, the CMP performs about 36% better than SEQ when we consider parallel regions. But when we consider the entire benchmark, the CMP performs about 6% worse than SEQ due to 52% coverage of non-parallelized regions. Similarly, *crafty, gcc, parser and vpr_place* suffer from poor sequential region performance.

On the other hand, the SMT configuration was able to dynamically reallocate its resources to exploit ILP when executing in sequential regions. Even though there is a slight slowdown in some benchmarks for SMT, the impact is much less when compared to CMP. For example, in *twolf* SMT-4 performs 27% better than SEQ while CMP-4-2MB slows down by about 6%, inspite of both achieving similar speedup inside parallel regions. Overall, SMT-4 performs about 36% better than SEQ if we consider only the parallel regions while its performance reduces to 23% when we consider the entire benchmark.

Low TLS parallelism inside parallelized regions: In benchmark *perlbmk*, as shown in Fig. 2(b), the loops selected have a poor iteration count leading to many threads being idle (indicated as *lack of threads*). Due to the limited parallelism available, the CMP did not get good performance, while SMT due to its dynamic resource allocation, uses the resources to extract ILP within the threads, resulting in a better performance than CMP. In benchmark *bzip2*,



(a) Speedup of entire program. (b) Normalized execution time breakdown of *all* compiler selected regions.

Fig. 2. Performance of SMT-4 and CMP-4-2MB configurations.

TABLE IV

COMPARISON OF THE IMPACT OF BENCHMARK BEHAVIORS ON THE PERFORMANCE OF SMT-TLS VS CMP-TLS.

Benchmark characteristics	Impact on CMP	Impact on SMT	Reasons
Large sequential regions	X	✓	SMT could use all resources to extract ILP inside sequential regions.
Low TLP inside parallel regions	X	✓	SMT effectively uses all its resources while many cores in CMP could be idle
High cache miss rates	✓	✓	Both can hide memory latency and speculative threads can prefetch data. SMT has more advantage due to shared L1.
Threads with a large working set	✓	X	SMT L1 cache overflows more often as it is shared by all threads, leading to more squashing.
Frequent mis-speculations	X	X	Mis-speculations wastes resources and affects non-speculative thread performance.

both SMT-4 and CMP-4-2MB have idle threads due to synchronization. But SMT-4 achieves better performance due to its better resource utilization. Similar effect can be seen in *mesa*, *gzip*, *vortex*, *vpr_route and* parser.

Large number of cache misses: In benchmark *equake* and in *mcf*, the SEQ configuration spends most of the execution time waiting for memory due to a large number of cache misses. Both CMP-4-2MB and SMT-4 are able to better hide the memory latency through sharing of the common working set. Such sharing of the working set allows some data needed by one thread to be *pre-fetched* by another thread. Due to the combined effect of parallelism and prefetching, both CMP-4-2MB and SMT-4 achieve good performance. Similarly, benchmarks *twolf* and *vpr_place* gain from good TLS parallelism and cache prefetching leading to performance gain for both SMT and CMP.

In SMT, both L1 cache and L2 cache are shared by all the threads, leading to better prefetching when compared to CMP where the threads share only the L2 cache. In *twolf and vpr_route*, SMT-4 performs better than CMP-4-2MB due to prefetching effect in L1 cache.

Size of threads: In benchmark *art*, the size of threads selected by the TLS compiler is quite large leading to speculative buffer overflow (part of *Others* in Fig. 2(b)). In the SMT-TLS [2] configuration, when there is buffer overflow, the younger speculative threads are preempted to make space for older speculative threads leading to extra squashes. However, as in *equake* and *mcf*, *art* has good cache prefetching effect leading to a good speedup inspite of its buffer overflow problem.

B. Power

To understand the power behavior of the two architectures, we compare the breakdown of dynamic power consumption in Fig. 3(a). The power consumption is normalized to the total power consumption of SEQ configuration. We used ideal clock gating (cc2) in the *Wattch* simulator to get dynamic power consumption.

Dynamic power is proportional to $\alpha C.V^2 f$, where α is the activity factor, C is the capacitance of the transistor, V the supply voltage, and f the frequency of the circuit. In our simulation, we kept V and f the same for all three configurations. So dynamic power differences among the three configurations are mainly due to the *a*ctivity factor or the *c*apacitance of the circuit.

Core complexity: The Superscalar uses the most complex core and has the highest C value while SMT core is also complex. But the CMP configuration uses smaller cores and, hence, has a smaller C value than that in Superscalar and SMT. The largest component of dynamic power, we call it the *window power*, combines the power consumption of function blocks related to out-of-order execution including RUU, LSQ, result bus, etc. The CMP configuration uses a smaller instruction window leading to lower window power consumption across all benchmarks. Similarly, it consumes less power in the cache since it uses a smaller cache than in other configurations.

Activity factor: SMT and CMP both execute the same parallel TLS code so their activity factor is very similar. However, SEQ runs the sequential code which does not have any special TLS instructions, leading to a smaller activity factor than SMT and CMP. Another factor which affects the activity is the amount of speculation. If a configuration suffers from frequent mis-speculations, it creates more speculative activities. As we saw in Fig. 2(b), the SMT configuration suffers from many *false* mis-speculations due to buffer overflow in *art*. These extra squashes leads to almost a 2X increase in dynamic power for SMT. Similar effect can be seen in *ammp*, *mesa*, *gzip*, *vortex*, *crafty* and *equake*. The SEQ has a more complex core than both SMT and CMP, and thus consumes higher power. But due to its

lower activity factor its power consumption is lower than SMT.

Extra hardware: The TLS architectures have extra power overhead due to the extra hardware needed to implement TLS. The extra hardware used by SMT is minimal, but CMP uses a common bus to connect the cores. The power overhead due to this common bus is significant, and not present in SEQ and SMT configurations.

Overall, due to the combined effect of complex cores and speculative wastage, SMT on average consumes about 32% more dynamic power than SEQ. CMP, due to its smaller cores, consumes about 10% less dynamic power than SEQ.

Total power: Total power consumption of the processor includes leakage/static power in addition to the dynamic power considered above. To get total power consumption, we use aggressive clock gating in *W*attch simulator (cc3).

The static power consumption depends on the program execution time and on the number of components that have leakage power (i.e. number of transistors). The SMT-4 configuration due to its lower execution time on average, consumes lesser static power than SEQ and CMP. While the CMP, due to its lower complexity can pack more resources in the same chip area. For example, the CMP-4-2MB uses two times the number of function units, RUU entries, etc. Due to the use of a larger number of components, the CMP has more leakage power than SMT.

In Fig. 3(b), we show both the dynamic and total power overhead of SMT and CMP over SEQ. In most benchmarks, due to its lower leakage power, the SMT is able to makeup for its increase in dynamic power. In *art*, the total power overhead of SMT is only 20% when compared to 159% overhead for dynamic power. Similar effect can be seen in *ammp*, *equake*, *vpr_route* and *vpr_place*. The register file in SMT-4 is 4 times larger than in SEQ to accomodate the 4 threads. This larger register file causes more leakage in benchmarks *gcc*, *perlbmk*, *mesa* and *parser*.

CMP consumes lower total power for *equake* and *art* due to its high speedup over SEQ. Total power overhead of CMP is higher than its dynamic power overhead in *perlbmk*, *parser*, *twolf*, *ammp* and *vpr_route*. For these benchmarks, CMP did not have a large performance gain and due to its larger resources it incurs more leakage power.

Overall, the CMP-4-2MB due to its lower performance suffers from 20% total power overhead when compared to SEQ while the SMT-4 suffers from 35% extra overhead due to its complexity. A summary of how the various factors affect power consumption in SMT and CMP is presented in Table V.

C. ED and ED^2

From the previous sections, we see that SMT and CMP have a very different behavior in power consumption and performance. To combine their effects we use *energy-delay product* (ED) and *energy-delay-squared product* (ED^2).

Fig. 4 shows the ED and ED^2 when we consider the entire program execution. As discussed before, when the sequential regions are included, the performance of CMP is

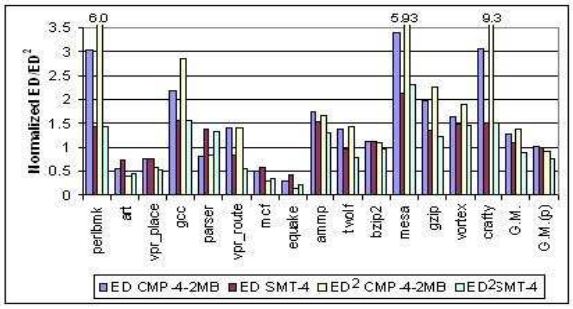


Fig. 4. ED and ED^2 of the entire program.

lower than that of SMT. Due to this slowdown in sequential regions, the ED of CMP is about 28% worse than that of SEQ and 37% worse in terms of ED^2. SMT-4 due to its large power overhead, performs 9% worse than SEQ in terms of ED but performs 11% better than SEQ in terms of ED^2 due to its better performance.

From the above discussion, it is clear that the SMT-4 configuration is more efficient in extracting TLS parallelism than the CMP-4-2MB configuration. In the next section, we consider different variations in the design space of CMP and SMT.

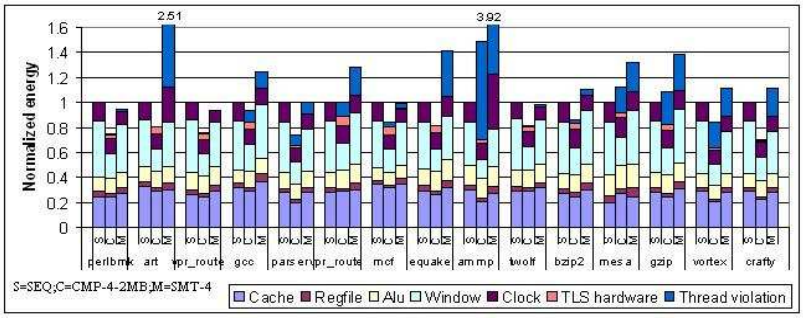
VI. Alternative Configurations

As we saw in previous section, the CMP based TLS performs worse than SMT based TLS due to its poor performance when executing in sequential regions. In this section, we study how the performance and power behavior change when we increase the core complexity to improve performance in sequential regions by varying key parameters such as the number of threads and L2 size.

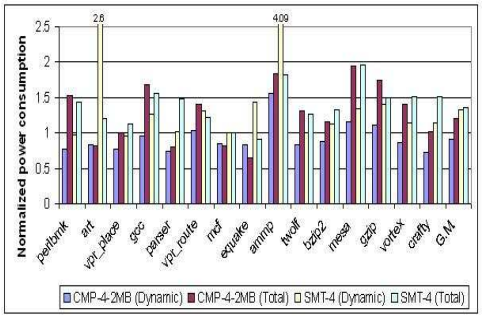
Impact of the number of threads: In Fig. 5 we compare the ED^2 of the 4-thread and 2-thread versions of both CMP and SMT architectures. Though the CMP-2-2MB performs better than CMP-4-2MB in sequential regions, it loses performance in parallel regions. Also the CMP-2-2MB cores are large and consume more power. On the average, due to its good performance in sequential region, the CMP-2-2MB has 22% lower ED^2 than CMP-4-2MB. But if we eliminate the lower performing benchmarks *perlbmk*, *mesa* and *crafty*, the ED^2 of CMP-4-2MB is 12% better than CMP-2-2MB and 9% better than SEQ (indicated by G.M.(p)).

In the case of SMT, one of the major causes for higher power consumption is the power wasted due to speculative execution (as shown in Fig. 3(a)). When we reduce the number of threads in SMT, this effect reduces and leads to large reduction in dynamic power consumption. Due to a large reduction in dynamic power, the SMT-2 has better ED^2 than SMT-4 in *perlbmk, parser, ammp, mesa, gzip, vortex and crafty*. While in other benchmarks SMT-4 has better ED^2 due to its superior performance in parallel region leading to overall 1% better ED^2 than SMT-2.

Impact of L2 size: Another possible design choice to improve sequential region performance is to reduce the L2 size, allowing the extra space to be used for larger cores. Fig. 6 compares the impact of the two configurations with a



(a) Normalized dynamic power consumption of the entire benchmark.



(b) Comparison of dynamic and static power overhead.

Fig. 3. Power consumption of SMT-4 and CMP-4-2MB configurations.

TABLE V

COMPARISON OF THE IMPACT OF VARIOUS FACTORS ON THE POWER CONSUMPTION OF SMT-TLS VS CMP-TLS.

Different factors	Impact on CMP	Impact on SMT	Reasons
Core complexity	✓	X	CMP with simpler cores consumes lesser dynamic power as seen in Fig. 3(a)
Execution time	X	✓	SMT has lower execution time than SEQ leading to lower leakage. But CMP slowsdown in some benchmarks leading to more leakage.
Threads causing overflow	✓	X	Overflow in SMT causes squashing, thus wasting more dynamic power (Fig. 3(a))
Number of transistors	X	✓	More transistors in CMP cause more leakage than in SMT.

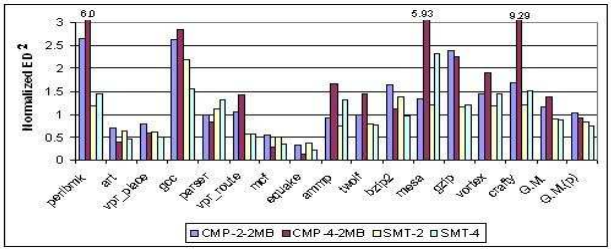


Fig. 5. Energy-delay-squared with the 2 and 4 threads.

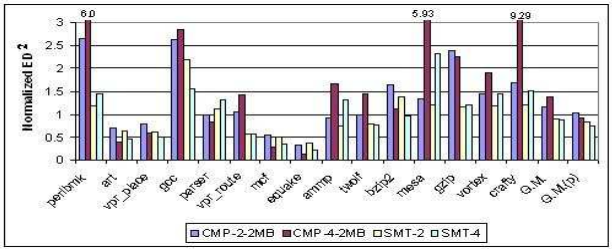


Fig. 6. Impact of larger cores on Energy-delay-squared of entire program.

smaller L2 size - CMP-4-1MB and CMP-2-1MB with CMP-4-2MB configuration.

CMP-4-1MB shows good improvement over CMP-4-2MB gaining about 10% speedup than the SEQ. But CMP-4-1MB consumes more power due to its larger cores, leading to increase in ED^2 (about 6% worse than CMP-4-2MB). Although the CMP-2-1MB configuration has a speedup up 2% over SEQ, its more complex cores leads to large increase in power consumption leading to 7% worse ED^2.

Impact of frequency: In our study, we had assumed the same clock frequency for all configurations. A simpler CMP core can be run at a higher frequency than in SEQ and SMT configurations. Though increasing frequency can lead to better performance, it leads to large increase in power consumption leading to worse ED^2.

Among the alternative design choices considered we found that reducing the number of cores in CMP (CMP-2-2MB) could lead to better ED^2 on average. But all the CMP configurations are still worse than the SMT-4 configuration in terms of ED^2.

VII. THERMAL BEHAVIOR

The Superscalar and the SMT-TLS architectures use complex cores with a large number of function units and large instruction window to exploit instruction-level parallelism or support the additional threads. These cores not only consume more energy, they can also generate thermal hotspots. On the other hand, the CMP-TLS architecture has distributed cores, and thus can potentially have smaller and less severe thermal hotspots. In this section, we analyze the thermal characteristics of three processor configurations—SEQ, SMT-4 and CMP-4-2MB.

The average and hotspot temperatures for each architecture are shown in Table VI. We have observed that the CMP-4-2MB configuration has the lowest average and hotspot temperatures, while the SMT-4 has the highest average and hotspot temperatures In terms of hotspot temperature, the CMP-4-2MB configuration is about 3.68 degrees lower than that of the SEQ configuration; while SMT-4 configuration is about 1.85 degrees higher than that of the SEQ configuration.

By observing the steady state temperature map for the SMT-4 and CMP-4-2MB configurations running `gcc`, which has the highest IPC among all benchmarks, we found that the main source of heat in both configurations is the register file (circled in Fig. 7). The temperature maps are shown in Figure 7. The activity level in the register file of each CMP core is lower than the activity level of the central register file in SMT-4, thus leading to lower hotspot temperature.

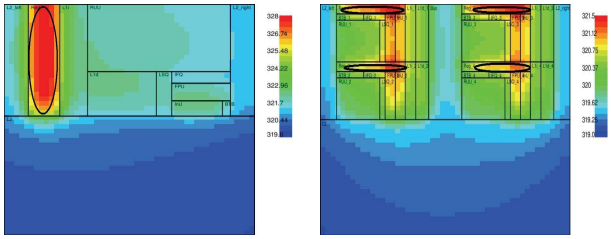
VIII. CONCLUSIONS

In this paper, we compared the performance, energy-delay-product and thermal effects of three architectures: Superscalar, SMT and CMP, while holding the die area constant. We have identified major issues in each of the architectures and found that the SMT-TLS is more suitable for TLS applications. From our results, we have shown that:

TABLE VI

THERMAL EFFECTS OF TLS ON THREE DIFFERENT ARCHITECTURES: SEQ, SMT-4 AND THE CMP-4-2MB IN DEGREE CELSIUS.

benchmark	SEQ		CMP-4-2MB		SMT-4	
	average	hotspot	average	hotspot	average	hotspot
perlbmk	61.21	66.38	59.66	62.9	61.89	68.12
art	57.55	65.92	58.48	62.16	60.07	67.97
vpr_place	60.17	65.96	60.64	62.27	61.62	67.99
gcc	60.33	66.02	59.14	62.33	61.28	67.96
parser	59.68	66.07	59.19	62.33	60.56	67.9
vpr_route	60.58	66.35	59.42	62.18	60.98	67.9
mcf	52.54	65.99	59.46	62.22	60.45	67.89
equake	56.71	65.93	59.22	62.16	60.05	67.89
ammp	59.17	66.02	62.18	59.52	61.15	68.01
twolf	60.15	65.93	60.02	62.17	61.42	67.93
bzip2	61.51	67.00	61.47	64.06	62.58	68.47
mesa	60.77	66.23	59.53	63.09	61.70	68.06
gzip	61.39	66.49	61.21	64.91	62.44	68.24
crafty	61.65	66.44	59.08	62.61	62.28	68.13
vortex	60.83	66.00	60.40	62.59	61.78	67.99
Mean	59.62	66.18	59.94	62.50	61.35	68.03



(a) SMT configuration. (b) CMP-4-2MB configuration.

Fig. 7. Thermal map for various configuration (running gcc). *Red* color indicates hottest regions.

- SMT-TLS can dynamically adjust its resources to achieve good TLS performance while not suffering significant slowdown in sequential code regions. The SMT-4 configuration achieves about 23% speedup over SEQ configuration.
- Nevertheless, the good performance of SMT-TLS comes at the cost of about 36% increase in power consumption when compared to Superscalar. But if we consider ED^2, the SMT-TLS outperforms both Supersclar and CMP-TLS architectures.
- The CMP-TLS architecture suffers due to poor sequential region performance. This can be improved by increasing the core complexity, but this increases power consumption. The CMP-2-2MB is the best CMP-TLS configuration which performs 26% worse than SMT-4 in terms of ED^2.
- The main disadvantage of SMT-TLS is that it creates more thermal stress than CMP-TLS due to its centralized register file.

Acknowledgements: This work is supported in part by a grant from National Science Foundation under EIA-0220021, a contract from Semiconductor Research Cooperation under SRC-2008-TJ-1819, and gift grants from Intel and IBM.

REFERENCES

[1] J. G. Steffan, C. B. Colohan, A. Zhai, and T. C. Mowry, "The stampede approach to thread-level speculation," in *ACM Trans. on Computer System*, vol. 23, August 2005, pp. 253–300.

[2] V. Packirisamy, S. Wang, A.Zhai, W.-C. Hsu, and P.-C. Yew, "Supporting speculative multithreading on simultaneous multithreaded processors," in *12th International Conference on High Performance Computing HiPC'2006*, Bengaluru. India, Dec. 2006.

[3] S. Kaxiras, G. J. Narlikar, A. D. Berenbaum, and Z. Hu, "Comparing power consumption of an smt and a cmp dsp for mobile phone workloads." in *CASES*, 2001.

[4] R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes, "The energy efficiency of cmp vs. smt for multimedia workloads." in *18th Annual ACM International Conference on Supercomputing*, 2004, pp. 196–206.

[5] Y. Li, D. Brooks, Z. Hu, and K. Skadron, "Performance, energy, and thermal considerations for smt and cmp architectures." in *11th International Symposium on High-Performance Computer Architecture (HPCA-11)*, 2005.

[6] J. Burns and J.-L. Gaudiot, "Area and system clock effects on smt/cmp throughput." *IEEE Trans. Computers*, vol. 54, no. 2, pp. 141–152, 2005.

[7] A. Zhai, C. B. Colohan, J. G. Steffan, and T. C. Mowry, "Compiler Optimization of Scalar Value Communication Between Speculative Threads," in *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, Oct 2002.

[8] A.Zhai, C.B.Colohan, J.G.Steffan, and T.C.Mowry, "Compiler Optimization of Memory-Resident Value Communication Between Speculative Threads," in *The 2004 International Symposium on Code Generation and Optimization*, Mar 2004.

[9] S. Wang, K. S. Yellajyosula, A. Zhai, and P.-C. Yew, "Loop Selection for Thread-Level Speculation," in *The 18th International Workshop on Languages and Compilers for Parallel Computing*, Oct 2005.

[10] I. Park, B. Falsafi, and T. Vijaykumar, "Implicitly-multithreaded processors," in *30th Annual International Symposium on Computer Architecture (ISCA '03)*, June 2003.

[11] J. Renau, K. Strauss, L. Ceze, W. Liu, S. R. Sarangi, J. Tuck, and J. Torrellas, "Energy-efficient thread-level speculation." *IEEE Micro*, vol. 26, no. 1, pp. 80–91, 2006.

[12] F. Warg and P. Stenström, "Dual-thread speculation: Two threads in the machine are worth eight in the bush," in *SBAC-PAD '06: Proceedings of the 18th International Symposium on Computer Architecture and High Performance Computing*, 2006, pp. 91–98.

[13] J. Lo, S. Eggers, J. Emer, H. Levy, R. Stamm, and D. Tullsen, "Converting Thread-Level Parallelism Into Instruction-Level Parallelism via Simultaneous Multithreading," pp. 322–354, Aug. 1997.

[14] S. Marc, K. Reiner, L. J. L., U. Theo, and V. Mateo, "Transistor count and chip-space estimation of simplescalar-based microprocessor models," in *Workshop on Complexity-Effective Design, in conjunction with the 28th International Symposium on Computer Architecture*, June 2001.

[15] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Reddi, and K. Hazelwood, "Pin: building customized program analysis tools with dynamic instrumentation," in *ACM SIGPLAN 05 Conference on Programming Language Design and Implementation (PLDI'05)*, June 2005.

[16] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *27th Annual International Symposium on Computer Architecture (ISCA '00)*, 2000.

[17] H. Wang, "Orion: A power-performance simulator for interconnection networks," 2002.

[18] W. Huang, K. Sankaranarayanan, R. J. Ribando, M. R. Stan, and K. Skadron, "An improved block-based thermal model in hotspot 4.0 with granularity considerations," in *Workshop on Duplicating, Deconstructing, and Debunking, in conjunction with the 34th International Symposium on Computer Architecture (ISCA)*, 2007.

[19] S. Wang, "Compiler Techniques for Thread-Level Speculation," Ph.D. dissertation, University of Minnesota, 2007.

[20] "Open research compiler for itanium processors," http://ipf-orc.sourceforge.net.

[21] T. Chen, J. Lin, X.Dai, W. Hsu, and P. Yew, "Data Dependence Profiling for Speculative Optimizations," in *Int'l Conf on Compiler Construction (CC)*, March 2004, pp. 57–62.

[22] E. Perelman, G. Hamerly, M. V. Biesbrouck, T. Sherwood, and B. Calder, "Using simpoint for accurate and efficient simulation," in *ACM SIGMETRICS Performance Evaluation Review*, 2003, pp. 318–319.