

# Performance, Energy, and Thermal Considerations for SMT and CMP Architectures

Yingmin Li<sup>†</sup>, David Brooks<sup>‡</sup>, Zhigang Hu<sup>††</sup>, Kevin Skadron<sup>†</sup>

<sup>†</sup> Dept. of Computer Science, University of Virginia <sup>††</sup> IBM T.J. Watson Research Center

<sup>‡</sup> Division of Engineering and Applied Sciences, Harvard University

{yingmin,skadron}@cs.virginia.edu, zhigangh@us.ibm.com, dbrooks@eecs.harvard.edu

## Abstract

*Simultaneous multithreading (SMT) and chip multiprocessing (CMP) both allow a chip to achieve greater throughput, but their relative energy-efficiency and thermal properties are still poorly understood. This paper uses Turandot, PowerTimer, and HotSpot to explore this design space for a POWER4/POWER5-like core. For an equal-area comparison with this style of core, we find CMP to be superior in terms of performance and energy-efficiency for CPU-bound benchmarks, but SMT to be superior for memory-bound benchmarks due to a larger L2 cache. Although both exhibit similar peak operating temperatures and thermal management overheads, the mechanism by which SMT and CMP heat up are quite different. More specifically, SMT heating is primarily caused by localized heating in certain key structures, CMP heating is mainly caused by the global impact of increased energy output. Because of this difference in heat up mechanism, we found that the best thermal management technique is also different for SMT and CMP. Indeed, non-DVS localized thermal-management can outperform DVS for SMT. Finally, we show that CMP and SMT will scale differently as the contribution of leakage power grows, with CMP suffering from higher leakage due to the second core's higher temperature and the exponential temperature-dependence of subthreshold leakage.*

## 1. Introduction

*Simultaneous multithreading (SMT)* [27] is a recent microarchitectural paradigm that has found industrial application [12, 18]. SMT allows instructions from multiple threads to be simultaneously fetched and executed in the same pipeline, thus amortizing the cost of many microarchitectural structures across more instructions per cycle. The promise of SMT is area-efficient throughput enhancement. But even though SMT has been shown energy efficient for most workloads [17, 21], the significant boost

in instructions per cycle (IPC) means increased power dissipation and possibly increased power density. Since the area increase reported for SMT execution is relatively small (10-20%), thermal behavior and cooling costs are major concerns.

*Chip multiprocessing (CMP)* [7] is another relatively new microarchitectural paradigm that has found industrial application [12, 14]. CMP instantiates multiple processor “cores” on a single die. Typically the cores each have private branch predictors and first-level caches and share a second-level, on-chip cache. For multi-threaded or multi-programmed workloads, CMP architectures amortize the cost of a die across two or more processors and allow data sharing within a common L2 cache. Like SMT, the promise of CMP is a boost in throughput. The replication of cores means that the area and power overhead to support extra threads is much greater with CMP than SMT. For a given die size, a single-core SMT chip will therefore support a larger L2 size than a multi-core chip. Yet the lack of execution contention between threads typically yields a much greater throughput for CMP than SMT [4, 7, 20]. A side effect is that each additional core on a chip dramatically increases its power dissipation, so thermal behavior and cooling costs are also major concerns for CMP.

Because both paradigms target increased throughput for multi-threaded and multi-programmed workloads, it is natural to compare them. This paper provides a thorough analysis of the performance benefits, energy efficiency, and thermal behavior of SMT and CMP in the context of a POWER4-like microarchitecture. In this research we assume POWER4-like cores with similar complexity for both SMT and CMP except for necessary SMT related hardware enhancements. Although reducing the CMP core complexity may improve the energy and thermal efficiency for CMP, it is cost effective to design a CMP processor by reusing an existing core. The POWER5 dual SMT core processor is an example of this design philosophy. We combine IBM's cycle-accurate Turandot [19] and PowerTimer [3, 9] performance and power modeling tools, modified to support both SMT and CMP, with University of

Virginia’s HotSpot thermal model [25]. Validation strategies for these tools have been discussed in [10, 17].

In general, for an SMT/CMP approach like IBM’s where the same base CPU organization is used, we find that CMP and SMT architectures perform quite differently for CPU and memory-bound applications. For CPU-bound applications, CMP outperforms SMT in terms of throughput and energy-efficiency, but also tends to run hotter, because the higher rate of work results in a higher rate of heat generation. The primary reason for CMP’s greater throughput is that it provides two entire processors’ worth of resources and the only contention is for L2. In contrast, SMT only increases the sizes of key pipeline structures and threads contend for these resources throughout the pipeline. On the other hand, for memory-bound applications, on an equal-area processor die, this situation is reversed, and SMT performs better, as the CMP processor suffers from a smaller amount of L2 cache.

We also find that the thermal profiles are quite different between CMP and SMT architectures. With the CMP architecture, the heating is primarily due to the global impact of higher energy output. For the SMT architecture, the heating is very localized, in part because of the higher utilization of certain key structures such as the register file. These different heating patterns are critical when we consider *dynamic thermal management* (DTM) strategies that seek to use runtime control to reduce hotspots. In general, we find that DTM strategies which target local structures are superior for SMT architectures and that global DTM strategies work better with CMP architectures.

The rest of the paper is organized as follows. In Section 2, we discuss the related work in comparing SMT and CMP processors from an energy-efficiency standpoint. Section 3 discusses the details of the performance, power, and temperature methodology that we utilize in this work, including our choice of L2 sizes to study. Section 4 discusses the baseline results for SMT and CMP architectures without DTM. Section 5 explores the more realistic case when microprocessors are DTM constrained and explores which strategies are best for CMP and SMT under performance and energy-constrained designs. Section 6 concludes the paper and discusses avenues for future research.

## 2. Related Work

There has been a burst of work in recent years to understand the energy efficiency of SMT processors. Li et al. [17] study the area overhead and energy efficiency of SMT in the context of a POWER4-like microarchitecture, and Seng et al. [21] study energy efficiency and several power-aware optimizations for a multithreaded Alpha processor. Sasanka et al. consider the energy-efficiency of SMT and CMP for multimedia workloads [20], and Kaxi-

ras et al. [13] do the same for mobile phone workloads on a digital signal processor. Like we do, these other studies find that SMT boosts performance substantially (by about 10–40% for SPEC workloads), and that the increase in throughput more than makes up for the higher rate of power dissipation, with a substantial net gain in energy efficiency.

For multithreaded and multiprogrammed workloads, CMP offers clear performance benefits. If contention for the second-level cache is not a problem, speedups are close to linear in the number of cores. Although energy efficiency of CMP organizations have been considered for specific embedded-system workloads, to our knowledge, the energy efficiency of CMP for high-performance cores and workloads has not been well explored. Sasanka et al. consider the energy-efficiency of SMT and CMP for multimedia workloads [20], and Kumar et al. [15] consider energy efficiency for a heterogeneous CMP core, but only for single-threaded workloads. Like we do, these other studies both find substantial energy benefits.

Other researchers have compared SMT and CMP. Sasanka et al., Kaxiras et al., Kumar et al. [16], Burns et al. [4], and Hammond et al. [7] all find that CMP offers a substantial performance advantage when there are enough independent threads to keep all cores occupied. This is generally true even when the CMP cores are simpler than the SMT core—assuming enough thread-level parallelism to take advantage of the CMP capability.

Several authors [4, 16, 20] also consider hybrids of SMT and CMP (e.g., two CMP cores, each supporting 2-way SMT), but with conflicting conclusions. They generally find a hybrid organization with N thread contexts inferior to CMP with N full cores, but to differing degrees. It is unclear to what extent these conclusions hold true specifically for memory-bound workloads. Since CMP seems superior to a hybrid organization, this work focuses only on purely 2-way SMT (one core) and 2-way CMP systems (one thread per core) in order to focus on the intrinsic advantages of each approach. While a study of the combined energy and thermal efficiency of hybrid CMP/SMT systems is interesting, we feel that it is beyond the scope of this paper: the incredibly complex design space described by [4, 16, 20] means that analyzing this configuration can easily occupy an entire paper by itself. In any case, understanding the combined energy and thermal efficiency of plain SMT and CMP systems is a prerequisite, and except for the work by Sasanka et al. and Kaxiras et al. for specialized workloads, we are not aware of any other work comparing the energy efficiency of SMT and CMP. Sasanka et al. find CMP to be much more energy efficient than SMT, while Kaxiras et al. find the reverse. The reason is that the Sasanka work uses separate programs which scale well with an increasing number of processors and can keep all processors occupied. In contrast, with the mo-

mobile phone workload of Kaxiras et al., not all threads are active all the time, and idle cores waste some energy. Instead, their SMT processor is based on a VLIW architecture and is wide enough to easily accommodate multiple threads when needed.

We are only aware of two other papers exploring thermal behavior of SMT and/or CMP. Heo et al. [8] look at a variety of ways to use redundant resources, including multiple cores, for migrating computation of a single thread to control hot spots, but find the overhead of core swapping is high. Donald and Martonosi [6] compare SMT and CMP and find that SMT produces more thermal stress than CMP. But, like many other studies comparing SMT and CMP, their analysis assumes that the cores of the CMP system are simpler and have lower bandwidth than the single-threaded and SMT processors, while we follow the pattern of the IBM POWER4/POWER5 series and assume that all three organizations offer the same issue bandwidth per core. Donald and Martonosi also consider a novel mechanism to cope with hotspots, by adding “white space” into these structures in a checkerboard fashion to increase their size and hopefully spread out the heat, but found that even a very fine-grained partitioning did not achieve the desired heat spreading. We adopt a similar idea for the register file, our key hotspot, but rather than increase its size, we throttle its occupancy. Simulations using an improved version of HotSpot in [11] suggest that sufficiently small structures will spread heat effectively.

### 3. Modeling Methodology

#### 3.1. Microarchitecture & Performance modeling

We use Turandot/PowerTimer to model an out-of-order, superscalar processor with resource configuration similar to current generation microprocessors. Table 3.1 describes the configuration of our baseline processor for the single-threaded design point (ST baseline). Note that, for efficient simulation of CMP systems, we currently need to use Turandot in trace-driven mode, so we cannot yet account for the impact of mis-speculated execution.

Processor Core	
Dispatch Rate	5 instructions per cycle
Reservation stations	mem/fix queue (2x20), fpq (2x5)
Functional Units	2 FXU, 2 FPU, 2 LSU, 1 BRU
Physical registers	80 GPR, 72 FPR
Branch predictor	16K-entry bimodal, 16K-entry gshare, 16K-entry selector, all with 1-bit entries
Memory Hierarchy	
L1 Dcache Size	32KB, 2-way, 128B blocks, 1-cycle latency
L1 Icache Size	64KB, 2-way, 128B blocks, 1-cycle latency
L2 I/D	2MB, 4-way LRU, 128B blocks, 9-cycle latency
Memory Latency	77 cycles

Table 1. Configuration of simulated processor.

SMT is modeled by duplicating data structures that correspond to duplicated resources and increasing the sizes

of those shared critical resources like the register file. Round-robin policy is used at various pipeline stages to decide which threads should go ahead. It will be our future work to try other scheduling policies like ICOUNT for our SMT performance model. More detail about these SMT enhancements can be found in [17].

We extended Turandot to model a CMP configuration. So far, only multi-programmed workloads without inter-thread synchronization are supported. This essentially consists of simulating two separate cores, except that cache and cache-bus conflicts in the shared L2 cache must be modeled, as they are important determinants of performance.

Performance comparison of different SMT or CMP configurations, or comparison of an SMT or CMP configuration against a single-threaded configuration, is difficult. Snavelly et al. [26] propose that

$$\text{SMT speedup} = \sum \frac{IPC_{SMT}[i]}{IPC_{nonSMT}[i]} \quad (1)$$

where  $IPC_{SMT}[i]$  is the IPC of just the  $i$ 'th thread during an SMT execution and  $IPC_{nonSMT}[i]$  is its IPC during single-threaded execution. This considers how each thread performs under SMT relative to its non-SMT performance, so we choose this metric for our speedup computations. All speedups are computed relative to the IPC of each workload on the baseline, non-SMT machine.

In contrast to evaluating performance, evaluating energy efficiency should use traditional, simple unweighted metrics.

#### 3.2. Benchmark Pairs

We use 15 SPEC2000 benchmarks as our single thread benchmarks. They are compiled by the *xlc* compiler with the -O3 option. First we used the Simpoint toolset [22] to get representative simulation points for 500-million-instruction simulation windows for each benchmark, then the trace generation tool generates the final static traces by skipping the number of instructions indicated by Simpoint and then simulating and capturing the following 500 million instructions.

We use pairs of single-thread benchmarks to form dual-thread SMT and CMP benchmarks. There are many possibilities for forming the pairs from these 15 benchmarks. We utilize the following methodology to form our pairs. First, we let each single thread benchmark combine with itself to form a pair. We also form several SMT and CMP benchmarks by combining different single thread benchmarks. We first categorize the single thread benchmarks into eight major categories: high IPC ( $> 0.9$ ) or low IPC ( $< 0.9$ ), high temperature (peak temperature  $> 82^\circ\text{C}$ ) or low temperature (peak temperature  $< 82^\circ\text{C}$ ), floating benchmark or integer benchmark as shown in Table 2.

We then form eighteen pairs of dual-thread benchmarks by selecting various combinations of benchmarks with

	gzip	mcf	eon	bzip2	crafty	vpr	ocl	parser
IPC	L	L	H	H	H	H	H	L
temperature	L	H	L	H	H	L	H	L
L2 miss ratio	L	H	L	L	L	L	L	L

	art	facerec	mgrid	swim	applu	mesa	ammp
IPC	L	H	H	L	L	H	L
temperature	H	H	H	H	H	L	H
L2 miss ratio	H	L	L	L	H	L	L

**Table 2. Categorization of benchmarks (integer benchmarks in the first table, floating point benchmarks in the second table)**

these characteristics. Note that our choice of memory-bound benchmarks was limited. This is a serious drawback to using SPEC for studies like this. The architecture community needs more benchmarks with a wider range of behaviors.

In the rest of the paper, we discuss our workloads in terms of those with high L2 cache miss ratio vs. those with low L2 cache miss ratio. When one benchmark in a pair has a high L2 cache miss ratio, we categorize that pair as a high L2 cache miss pair.

### 3.3. Power Model

PowerTimer differs from existing academic microarchitectural power-performance simulators primarily in energy-model formation [3, 9]. The base energy-models are derived from circuit-level power analysis that has been performed on structures in a current, high-performance PowerPC processor. This analysis has been performed at the macro level, and in general, multiple macros will combine to form a microarchitectural level structures corresponding to units within our performance model. PowerTimer models over 60 microarchitectural structures which are defined by over 400 macro-level power equations. If not mentioned, we assume uniform leakage power density for all the units on the chip if they have the same temperature. Leakage power is estimated based on a formula derived by curve fitting with the ITRS data [23]. Leakage power of one unit depends on the area and temperature of that unit. Incorporating more accurate leakage power models will improve the accuracy of the results, especially for future technologies—an important area for future work.

### 3.4. Temperature Model

To model operating temperature, we use the newly released HotSpot 2.0 (<http://lava.cs.virginia.edu/HotSpot>), which accounts for the important effects of the thermal interface material (TIM) between the die and heat spreader and has been validated against a test chip [10].

HotSpot models temperature using a circuit of thermal resistances and capacitances that are derived from the layout of microarchitecture units. The thermal package that

is modeled consists of the die-to-spreader TIM (thickness 0.05mm), the heat spreader (thickness 1mm), another TIM, the heat sink (thickness 6.9mm), and a fan. Removal of heat from the package via airflow takes place by convection and is modeled using a single, equivalent thermal resistance of 0.8K/W. This assumes the fan speed and the ambient temperature inside the computer “box” (40°C) are constant, both of which are true for the time scales over which our benchmarks are simulated.

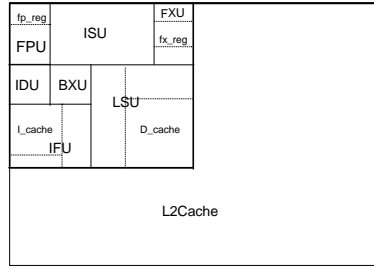
Due to lateral heat spreading, thermal behavior is sensitive to the layout of the microarchitecture units. We use the floorplans shown in Figure 1, which have been derived by inspection from the die photo of the POWER5 in [5]. Note that Figure 1 only shows floorplans for the single-threaded and CMP chips. The SMT floorplan is identical to the single-threaded case, except that the increase in resources to accommodate SMT makes the core 12% larger. (This is small enough—a few percent of the total chip area—that we take the impact on L2 size for SMT to be negligible.)

According to [5], the POWER5 offers 24 sensors on chip. Accordingly, we assume it is reasonable to provide at least one temperature sensor for each microarchitecture block in the floorplan, and that these sensors can be placed reasonably close to each block’s hot spot, or that data fusion among multiple sensors can achieve the same effect. We also assume that averaging and data fusion allow dynamic noise to be ignored, and that offset errors can be removed by calibration [1]. We sample the temperature every 100k cycles and set our DTM experiments’ thermal emergency threshold at 83°C. This threshold is carefully chosen so for single thread single core architecture it will normally lead to less than 5% performance loss due to DTM control. At the beginning of the simulation, we set the steady state temperature for each unit as the initial temperature so the whole simulation’s thermal output will be meaningful. For DTM experiments, the initial temperature is set as the smaller value of the steady state temperature without DTM and the thermal emergency threshold which is 83°C in our DTM experiments.

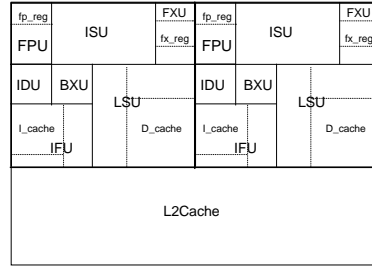
### 3.5. Chip Die Area and L2 Cache Size Selection

Before performing detailed equal-area comparisons between CMP and SMT architectures, it is important to carefully select appropriate L2 cache sizes for the baseline machines. Because the core area stays fixed in our experiments, the number of cores and L2 cache size determines the total chip die area. In particular, because the CMP machine requires additional chip area for the second core, the L2 cache size must be smaller to achieve equivalent die area. In this study, the additional CMP core roughly equals 1MB of L2 cache.

In the 2004-2005 timeframe, mainstream desktop and server microprocessors include aggressive, out-of-order processor cores coupled with 512KB to 2MB of on-chip



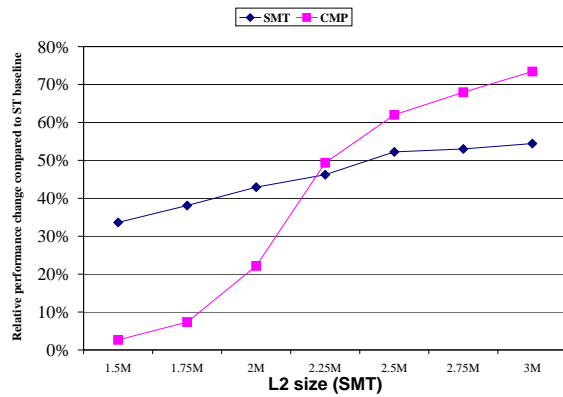
(a) ST & SMT



(b) CMP

**Figure 1. Floorplans used for thermal simulation. The SMT core is 12% larger than the ST core shown above.**

L2 cache. Our experiments indicate that for very large L2 cache sizes and typical desktop and workstation applications (SPEC2000), most benchmarks will fit in the cache for both the SMT and CMP machines. But for a fixed number of cores, Figure 2 shows that as die size is reduced, SMT eventually performs better than CMP for memory-bound benchmarks. This is because a core occupies about 1 MB’s worth of space, so SMT’s L2 sizes are 1 MB larger than CMP’s. Given constraints on chip area, it is likely that there will always be certain memory-bound workloads that will perform better with SMT than with CMP. Recognizing this tradeoff, we set the L2 cache at 1MB for CMP and at 2MB for SMT for our baseline study and discuss where appropriate how these choices impact our conclusions.



**Figure 2. Performance of SMT and CMP for memory-bound benchmarks (the categorization is done with 2MB L2 cache size for ST) with different L2 cache size.**

## 4. Baseline Results

In this section, we discuss the performance, energy, and temperature implications of SMT and CMP designs *without* dynamic thermal management. In the next section, we consider thermally limited designs.

When we compare the three architectures (ST, SMT, and CMP), we hold the chip area as a constant at 210

mm<sup>2</sup> including the on-chip level two cache. This means CMP will have the smallest L2 cache, since its core area is largest among the three. In our experiment, the L2 cache sizes for ST, SMT, and CMP are 2MB, 2MB, and 1MB respectively. Because the SMT core is only 12% larger than the ST core, we use 2MB for both.

Because we find the conclusions are quite different for workloads with high L2 miss rate vs. those with lower miss rates, we normally report results for these categories separately.

### 4.1. SMT and CMP Performance and Energy

Figure 3 breaks down the performance benefits and energy efficiency of SMT and CMP for our POWER4-like microarchitecture. The results in this figure are divided into two classes of benchmarks – those with relatively low L2 miss rates (left) and those with high L2 cache miss rates (right). This figure shows that CMP dramatically outperforms SMT for workloads with low to modest L2 miss rates, with CMP boosting throughput by 87% compared to only 26% for SMT. But the CMP chip has only half the L2 cache as SMT, and for workloads with high L2 miss rate, CMP only affords a throughput benefit of 22% while SMT achieves a 42% improvement.

The power and energy overheads demonstrated in Figure 3 are also enlightening. The power overhead of SMT is 38–46%. The main reasons for the SMT power growth are the increased resources that SMT requires (e.g. replicated architected registers), the increased resources that are needed to reduce new bottlenecks (e.g. additional physical registers), and the increased utilization due to additional simultaneous instruction throughput [17]. The power increase due to CMP is even more substantial: 93% for low-L2-miss-rate workloads and 71% for the high-miss-rate workloads. In this case the additional power is due to the addition of an entire second processor. The only reason the power does not double is that L2 conflicts between the two cores lead to stalls where clock gating is engaged, and this explains the lower power overhead of the L2-bound workloads.

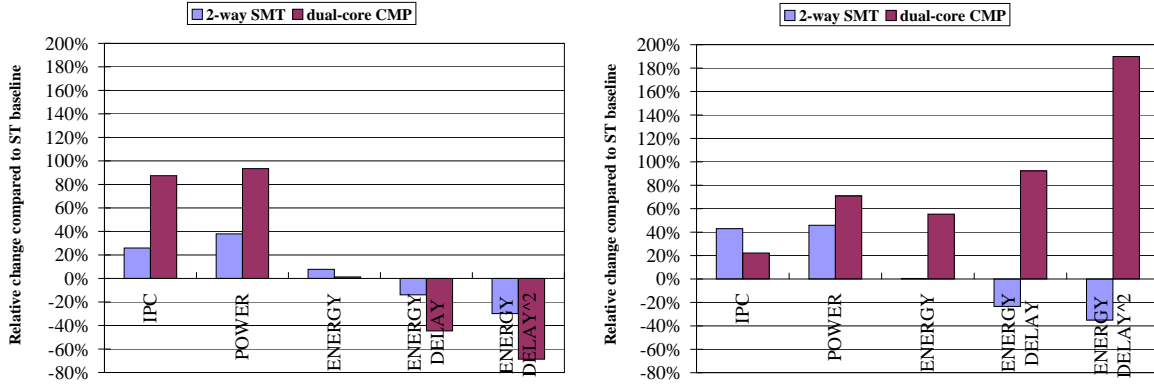


Figure 3. Performance and Energy efficiency of SMT and CMP compared to ST, for low L2 cache miss workloads (left) and high L2 cache miss workloads (right).

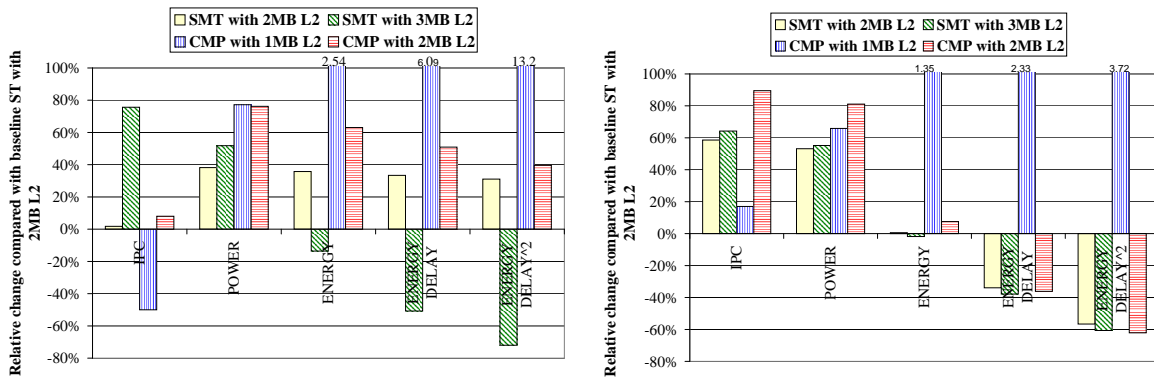


Figure 4. Performance and Energy efficiency of SMT and CMP compared to ST as L2 size changes. On the left are results for a benchmark (mcf+mcf) which is memory bound for all L2 configurations shown. On the right are results for a benchmark (mcf+vpvr) with ceases to be memory-bound once L2 size changes from 1MB to 2MB for CMP.

Combining these two effects with the energy-delay-squared metric ( $ED^2$ ) [28], we see that CMP is by far the most energy-efficient organization for benchmarks with reasonable L2 miss rates, while SMT is by far the most energy-efficient for those with high miss rates. Indeed, for L2-bound workloads, from the standpoint of  $ED^2$ , a single-threaded chip would be preferable to CMP, even though the single-threaded chip cannot run threads in parallel. Of course, this is at least in part due to the reduced L2 on the CMP chip.

When we increase L2 cache size, some benchmarks that had previously been memory bound now fit better in the L2 cache, and thus need to be categorized as low L2 miss rate benchmarks. Figure 4 illustrates the consequences. The graph on the right shows how mcf+vpvr ceases to be memory bound when we increase the L2 cache sizes by 1 MB (SMT from 2MB to 3MB and CMP from 1MB to 2MB). With smaller L2 cache size and high cache miss ratio, the program is memory-bound and SMT is better in terms of performance and energy efficiency. With larger L2 size and low cache miss ratio, the program is no longer memory bound and CMP is better. Of course, for any L2 size, some

applications' working set will not fit, and these benchmarks will remain memory bound. The left-hand graph in Figure 4 illustrates that SMT is superior for memory-bound benchmarks.

To summarize, once benchmarks have been categorized for an L2 size under study, the qualitative trends we report for the compute-bound and memory-bound categories seem to hold.

## 4.2. SMT and CMP Temperature

Figure 5 compares the maximum measured temperature for several different microprocessor configurations. We see that the single-threaded core has a maximum temperature of nearly  $82^\circ C$ . When we consider the SMT processor, the temperature increases around 7 degrees and for the CMP processor the increase is around 8.5 degrees.

With such a small difference in temperature, it is difficult to conclude that either SMT or CMP is superior from a temperature standpoint. In fact, if we rotate one of the CMP cores by 180 degrees, so the relatively cool IFU of core 1 is adjacent to the hot FXU of core 0, the maximum

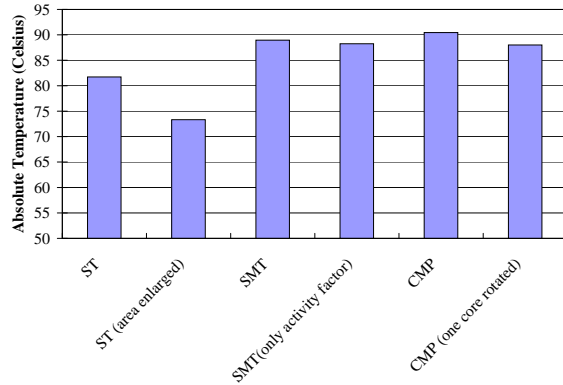


Figure 5. Temperature of SMT and CMP vs. ST.

CMP processor temperature will drop by around 2 degrees, which makes it slightly cooler than the SMT processor.

Despite the fact that the SMT and CMP processors have relatively similar absolute temperature ratings, the reasons for the SMT and CMP hotspots are quite different. In order to better understand underlying reasons behind the temperature increases in these machines, we have performed additional experiments to isolate the important effects.

We have taken the SMT core and only scaled the power dissipation with increased utilization (omitting the increased power dissipation due to increased resources and leaving the area constant). From Figure 5 we can see that the SMT temperature will rise to nearly the same level as when all three factors are included. This makes sense when we consider that the *unconstrained power density* of most of the scaled structures in the SMT processor (e.g. register files and queues) will likely be relatively constant because the power and area will both increase with the SMT processor, and in this case the utilization increase becomes the key for SMT hotspots. From this we can conclude that for the SMT processor, the temperature hotspots are largely due to the higher utilization factor of certain structures like the integer register file.

The reasoning behind the increase in temperature for the CMP machine is quite different. For the CMP machine, the utilization of each individual core is nearly the same as for the single-thread architecture. However, on the same die area we have now integrated two cores and the total power of the chip nearly doubles (as we saw in Figure 3) and hence the total amount of heat being generated nearly doubles. Because of the large chip-level energy consumption, the CMP processor heats up the TIM, heat spreader, and heat sink, thus raising the temperature of the overall chip. Thus the increased temperature of the CMP processor is due to a global heating effect, quite the opposite of the SMT processor's localized utilization increase. This fundamental difference in thermal trends will lead to substantial differences in thermal trends as we consider future technologies and advanced dynamic thermal management techniques.

### 4.3. Impact of Technology Trends

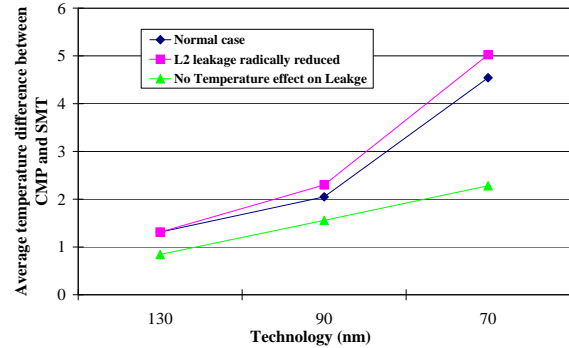


Figure 6. Temperature Difference between CMP and SMT for different technologies.

As we move towards the 65nm and 45nm technology nodes, there is universal agreement that leakage power dissipation will become a substantial fraction of the overall chip power. Because of the basic difference in the reasons for increased thermal heating between the SMT and CMP processors, we expect that these processors will scale differently as leakage power becomes a more substantial portion of total chip power.

Figure 6 shows the impact of technology scaling on the temperature of SMT and CMP processors. In this figure, we show the difference in absolute temperature between the CMP and SMT core for three generations of leakage (roughly corresponding to 130nm, 90nm, and 70nm technologies). As the we project towards future technologies, there are several important trends to note. The most important trend is that the temperature difference between the CMP machine (hotter) and SMT machine (cooler) increases from 1.5 degrees with our baseline leakage model to nearly 5 degrees with the most leaky technology. The first reason for this trend is that the increased utilization of the SMT core becomes muted by higher leakage. The second reason is that the SMT machine's larger L2 cache tends to be much cooler than the second CMP core. This, coupled with the exponential temperature dependence of subthreshold leakage on temperature, causes the CMP processor's power to increase more than the SMT processor. This aggravates the CMP processor's global heat up effect. From Figure 6, we can see that if we remove the temperature dependence of leakage in our model, the temperature difference between the CMP and SMT machine grows much less quickly. Figure 6 also shows how the trend is amplified when we consider the case where aggressive leakage control is applied to the L2 cache (perhaps through high-Vt transistors). In this case, the SMT processor is favored because a larger piece of the chip is eligible for this optimization.

## 5. Aggressive DTM constrained designs

To reduce packaging cost, current processors are usually designed to sustain the thermal requirement of typical workloads, and engage some dynamic thermal management techniques when temperature exceeds the design set point. Because SMT and CMP dissipate more power and run hotter, a more accurate comparison of their relative benefits requires data on their cooling costs, whether those costs are monetary in terms of more expensive packaging, or performance losses from DTM. This section explores the impact of different DTM strategies upon the performance and energy efficiency of SMT and CMP, and how these DTM results explain the different thermal behavior of these two organizations.

It is important to note that peak temperature is not indicative of cooling costs. A benchmark with short periods of very high temperature, separated by long periods of cooler operation, may incur low performance overhead from DTM, while a benchmark with more moderate but sustained thermal stress may engage DTM often or continuously.

To make an equal comparison of DTM performance among single-threaded, SMT, and CMP chips, we continue to use the same thermal package for all three configurations (see Section 3).

### 5.1. DTM Techniques

We implemented four DTM strategies in this paper:

- **Dynamic voltage scaling (DVS):** DVS cuts voltage and frequency in response to thermal violations and restores the high voltage and frequency when the temperature drops below the trigger threshold. The low voltage is always the same, regardless of the severity of thermal stress; this was shown in [24] to be just as effective as using multiple V/F pairs and a controller. For these workloads, we found that a voltage of 0.87 (79% of nominal) and frequency of 1.03GHz (77% of nominal) were always sufficient to eliminate thermal violations. Because there is not yet a consensus on the overhead associated with switching voltage and frequency, we test both 10 and 20  $\mu$ s stall times for each change in the DVS setting.
- **Fetch-throttling:** Fetch-throttling limits how often the fetch stage is allowed to proceed, which reduces activity factors throughout the pipeline. The duty cycle is set by a feedback controller.
- **Rename-throttling:** Rename throttling limits the number of instructions renamed each cycle. Depending on which register file is hotter with the outcome of the previous sampling period, either floating-point register renaming or integer register renaming will be throttled. This reduces the rate at which a thread can

allocate new registers in whichever register file has overheated, and is thus more localized in effect than fetch throttling. But if the throttling is severe enough, this has the side effect of slowing down the thread that is causing the hot spot. This can degenerate to fetch throttling, but when it is the FP register file being throttled, the slowdown can be valuable for mixed FP-integer workloads by helping to regulate resource use between the two threads.

- **Register-file occupancy-throttling:** We find the register file is usually the hottest spot of the whole chip, and its power is proportional to the occupancy. One way to reduce the power of the register file is to limit the number of register entries to a fraction of the full size. To distribute the power density, we propose to interleave the on and off registers, so that the heat can be more evenly spreaded across the whole register file. It is important to note that our modeling of this technique is idealistic, assuming that the reduction in power density across the register file is proportional to the number of registers that have been turned off. This assumes an ideal interleaving and ideal heat spreading and neglects power dissipation in the wiring, which will not be affected with occupancy throttling. This technique is included to demonstrate the potential of value of directly reducing power density in the structure that is overheating, rather than reducing activity in the whole chip.

By limiting the resources available to the processor, all these policies will cause the processor to slow down, thus consuming less power and finally cooling down to below the thermal trigger level. DVS has the added advantage that reducing voltage further reduces power density; since  $P \propto V^2 f$ , DVS provides roughly a cubic reduction in heat dissipation relative to performance loss,<sup>1</sup> while the other techniques are linear. But the other techniques may be able to hide some of their performance loss with instruction-level parallelism. Of the three policies, fetch-throttling has more of a global effect over the whole chip by throttling the front end. Register-file occupancy throttling targets the specific hot units (the integer register file or the floating point register file) most directly and thus is the most localized in effect. This may incur less performance loss but also may realize less cooling. Rename throttling is typically more localized than fetch throttling and less so than register-file throttling.

DVS's cubic advantage is appealing, but as operating voltages continue to scale down, it becomes more difficult to implement a low voltage that adequately cuts temperature while providing correct behavior and reasonable frequency. Another concern with DVS is the need to validate products for two voltages rather than one. Finally, our

<sup>1</sup> This is only an approximate relationship; our experiments derive the actual V-f relationship from ITRS data [23].

assumption that both frequency and voltage can change in 10–20  $\mu\text{s}$  may be optimistic. If voltage and frequency must change gradually to avoid circuit noise, the latency to achieve adequate temperature reduction may be prohibitively long.

Our register-occupancy throttling is limited to register files based on a latch-and-mux design. Power dissipation in SRAM-based designs is likely to be much more heavily dominated by the decoders, sense amplifiers, and word and bit lines—another interesting area for future work. Furthermore, our technique may be idealistic, because it assumes that reducing register file occupancy uniformly reduces power density, when in fact those registers that remain active will retain the same power dissipation. But this does not mean that the temperature of active registers remains unchanged, because neighboring areas of lower power density can help active registers to spread their heat. Whether a register is small enough to spread enough heat laterally is an open question and requires further analysis. However, results in [11] using HotSpot 2.0 suggest that, below about 0.2–0.25 mm and for a 0.5mm die with a typical high-performance package, the ratio of vertical to lateral thermal resistance is so high that heat spreads out very quickly, without raising the localized temperature. This result differs from the findings of [6], who used HotSpot 1.0 to find that much smaller sizes are needed to spread heat. But HotSpot 1.0 omits the TIM’s very high thermal resistance and performs less detailed thermal modeling of heat flow in the package. Clearly the granularity at which spreading dominates, and alternative layouts and organizations which can reduce hotspots, is an important area requiring further research. But almost all prior DTM research has focused on global techniques like fetch gating, voltage-based techniques, or completely idling the hot unit, all of which suffer from significant overheads. What is needed are techniques that can reduce power density *in situ*, without introducing stalls that propagate all the way up the pipeline. Our register-occupancy throttling illustrates that such an approach offers major potential benefits, and that further research in this direction is required.

## 5.2. DTM Results: Performance

For many traditional computing design scenarios, performance is the most critical parameter, and designers primarily care about power dissipation and thermal considerations because of thermal limits. In these cases, designers would like to optimize performance under thermal constraints. These include systems such as traditional PC desktops and certain high-performance server environments where energy utility costs are not critical.

To evaluate architectures viable for these situations, Figure 7 shows performance of SMT and CMP architectures with different DTM schemes. As we observed in the previous section, the results are again dependent on

whether the workloads have high L2 miss ratio. For workloads with low or moderate miss ratios, CMP always gives the best performance, regardless of which DTM technique is used. On the other hand, for workloads that are mostly memory bound, SMT always gives better performance than CMP or ST.

When comparing the DTM techniques, we found that DVS10, the DVS scheme assuming an optimistic 10  $\mu\text{s}$  voltage switch time, usually gives very good performance. This is because DVS is very efficient at reducing chip-wide power consumption, thus bringing chip-wide temperature down very quickly and allowing the chip to quickly revert back to the highest frequency. When assuming a more pessimistic switching time of 20  $\mu\text{s}$ , the performance of DVS degrades a lot, but is still among the best of the the DTM schemes. However, in a system where energy consumption is not a primary concern, DVS may not be available due to the high implementation cost, while the relatively easier-to-implement throttling mechanisms are available. In the rest of this section, we mainly focus on the behavior of the non-DVS techniques.

Looking at the low L2 miss workloads (Figure 7, left) and the high L2 miss workloads (Figure 7, right), we find that SMT and CMP diverge with regards to the optimal throttling scheme. For CMP, fetch-throttling and register-occupancy throttling work equally well, and both outperform local rename-throttling. For SMT, register throttling is the best performing throttling scheme, followed by rename-throttling and global fetch-throttling. In fact, for SMT running high L2 miss workloads, the local register occupancy throttling performs better than all of the other DTM techniques including DVS.

The relative effectiveness of the DTM techniques illustrates the different heating mechanisms of CMP and SMT, with heating in the CMP chip a more global phenomenon, and heating in the SMT chip localized to key hotspot structures. For example, by directly resizing the occupancy of the register file, register-throttling is very effective at reducing the localized power density of the register file, and bringing down the temperature of the register file. In other words, the match-up between the mechanism of register-throttling and the inherent heat-up mechanism makes register-throttling the most effective DTM scheme for SMT. On the other hand, CMP mainly suffers from the global heat up effects due to the increased power consumption of the two cores. Thus global DTM schemes that quickly reduce total power of the whole chip perform best for CMP. We have found that this conclusion remains unchanged when increasing the L2 cache size to 2MB for CMP.

## 5.3. DTM Results: Energy

In many emerging high-performance computing environments, designers must optimize for raw performance

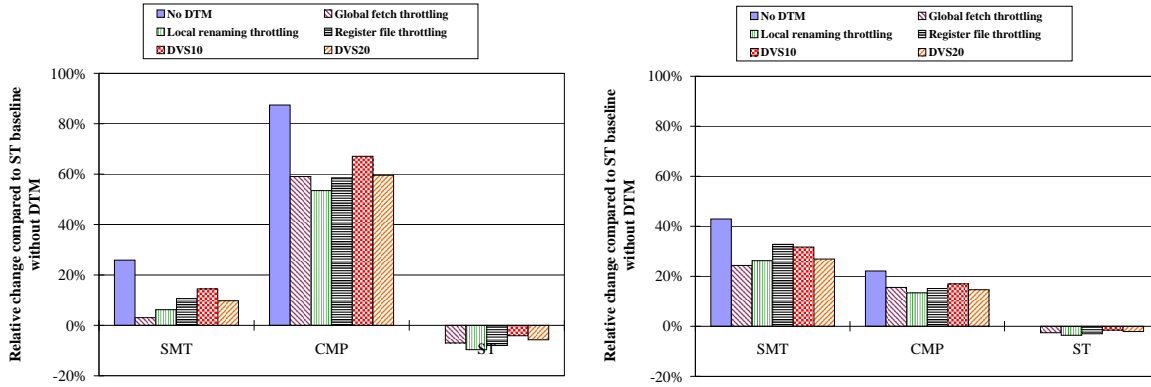


Figure 7. Performance of SMT and CMP vs. ST. with different DTM policies, all with threshold temperature of 83°C. Workloads with low L2 cache miss rate are shown on the left. Workloads with high L2 cache miss rate are shown on the right.

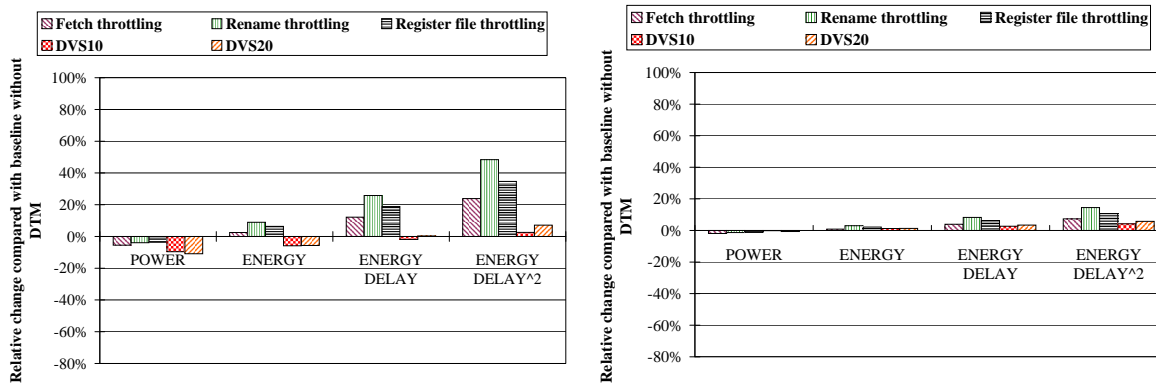


Figure 8. Energy-efficiency metrics of ST with DTM, compared to ST baseline without DTM, for low-L2-miss-rate workloads (left) and high-L2-miss-rate workloads (right).

under thermal packaging constraints, but energy consumption is also a critical design criteria for battery life or for energy utility costs. Examples of these systems are high-performance mobile laptops and servers designed for throughput oriented data centers like the Google cluster architecture [2].

In this scenario, designers often care about joint power-performance system metrics after DTM techniques have been applied. Figure 8 through Figure 10 shows the power and power-performance metrics (energy, energy-delay, and energy-delay<sup>2</sup>) for the ST, SMT, and CMP architectures after applying the DTM techniques. All of the results in these figures are compared against the baseline ST machine without DTM. From these figure, we see that the dominant trend is that global DTM techniques, in particular DVS, tend to have superior energy-efficiency compared to the local techniques for most configurations. This is true because the global nature of the DTM mechanism means that a larger portion of the chip will be cooled, resulting in a larger savings. This is especially obvious for the DVS mechanism, because DVS’s cubic power savings is significantly

higher than the power savings that the throttling techniques provide. The two local thermal management techniques, rename and register file throttling, do not contribute to a large power savings while enabled, as these techniques are designed to target specific temperature hotspots and thus have very little impact on global power dissipation. However, from an energy-efficiency point of view, local techniques can be competitive because in some cases they offer better performance than global schemes.

Figure 8 shows the results for the ST machine. Because DTM is rarely engaged for the ST architecture, there is a relatively small power overhead for these benchmarks. These ST results provide a baseline to decide whether SMT and CMP are still energy-efficient after DTM techniques are applied.

From Figure 9 we can see that the SMT architecture is superior to the ST architecture for all DTM techniques except for rename throttling. As expected, the DVS techniques perform quite well, although with high-L2 miss rate benchmarks register file throttling, due to performance advantages, does nearly as well as DVS for ED<sup>2</sup>.

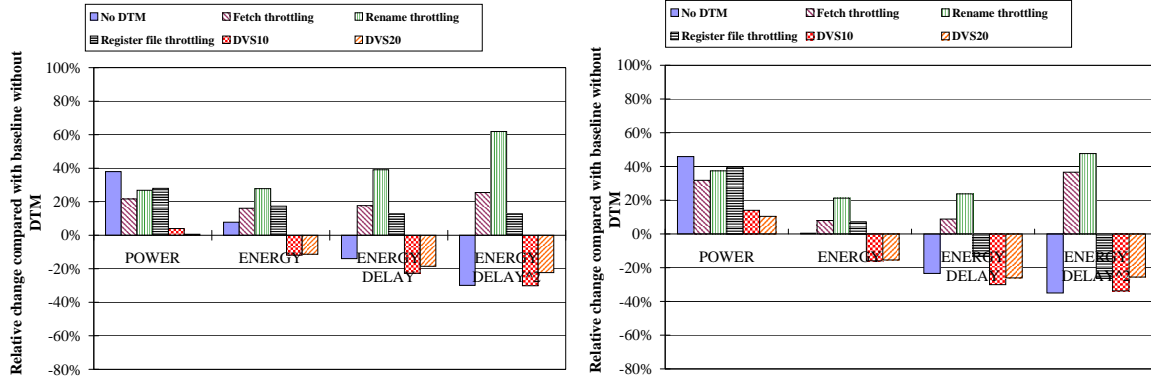


Figure 9. Energy-efficiency metrics of SMT with DTM, compared to ST baseline without DTM, for low-L2-miss-rate benchmarks (left) and high-L2-miss-rate benchmarks (right).

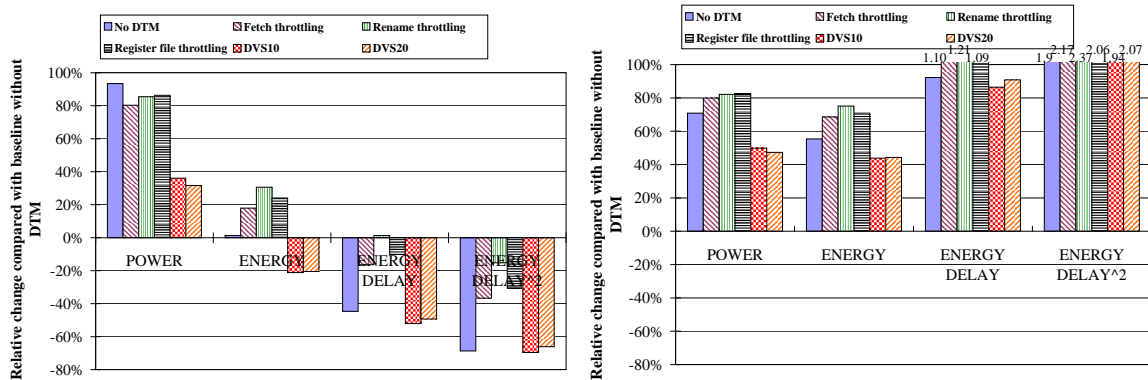


Figure 10. Energy-efficiency metrics of CMP with DTM, compared to ST baseline without DTM, for low-L2-miss-rate benchmarks (left) and high-L2-miss-rate benchmarks (right).

Figure 10 allows us to compare CMP to the ST and SMT machines for energy-efficiency after applying DTM. When comparing CMP and SMT, we see that for the low-L2 miss rate benchmarks, the CMP architecture is always superior to the SMT architecture for all DTM configurations. In general, the local DTM techniques do not perform as well for CMP as they did for SMT. We see the exact opposite behavior when considering high-L2 miss rate benchmarks. In looking at the comparison between SMT and CMP architectures, we see that for the high-L2 miss rate benchmarks, CMP is not energy-efficient relative to *either* the baseline ST machine or the SMT machine—even with the DVS thermal management technique.

In conclusion, we find that for many, but not all configurations, global DVS schemes tend to have the advantage when energy-efficiency is an important metric. The results do suggest that there could be room for more intelligent localized DTM schemes to eliminate individual hotspots in SMT processors, because in some cases the performance benefits could be significant enough to beat out global DVS schemes.

## 6. Future Work and Conclusions

This paper provides an in-depth analysis of the performance, energy, and thermal issues associated with simultaneous multithreading and chip-multiprocessors. Our broad conclusions can be summarized as follows:

- CMP and SMT exhibit similar operating *temperatures* within current generation process technologies, but the heating *behaviors* are quite different. SMT heating is primarily caused by localized heating within certain key microarchitectural structures such as the register file, due to increased utilization. CMP heating is primarily caused by the global impact of increased energy output.
- In future process technologies in which leakage power is a significant percentage of the overall chip power CMP machines will generally be hotter than SMT machines. For the SMT architecture, this is primarily due to the fact that the increased SMT utilization is overshadowed by additional leakage power. With the CMP machine, replacing the relatively cool L2 cache with a sec-

ond core causes additional leakage power due to the temperature-dependent component of subthreshold leakage.

- For the organizations we studied, CMP machines offer significantly more throughput than SMT machines for CPU-bound applications, and this leads to significant energy-efficiency savings despite a substantial (80%+) increase in power dissipation. However, in our equal-area comparisons between SMT and CMP, the loss of L2 cache hurts the performance of CMP for L2-bound applications, and SMT is able to exploit significant thread-level parallelism. From an energy standpoint, the CMP machine's additional performance is no longer able to make up for the increased power output and energy-efficiency becomes negative.
- CMP and SMT cores tend to perform better with different DTM techniques. In general, in performance-oriented systems, localized DTM techniques work better for SMT cores and global DTM techniques work better for CMP cores. For energy-oriented systems, global DVS thermal management techniques offer significant energy savings. However, the performance benefits of localized DTM make these techniques competitive for techniques for energy-oriented SMT machines.

In our future work, we hope to tackle the challenging problem of considering significantly larger amounts of thread-level parallelism and considering hybrids between CMP and SMT cores. We will also explore the impact of varying core complexity on the performance of SMT and CMP, and explore a wider range of design options, like SMT fetch policies. There is also significant opportunity to explore tradeoffs between exploiting TLP and core-level ILP from energy and thermal standpoints. Finally, we also would like to explore server-oriented workloads which are likely to contain characteristics that are most similar to the memory-bound benchmarks from this study.

## Acknowledgments

This work was funded in part by the National Science Foundation under grant nos. CCR-0133634, EIA-0224434, a Faculty Partnership Award from IBM T.J.Watson, and an Excellence Award from the University of Virginia Fund for Excellence in Science and Technology. This work was partly done at IBM T.J.Watson research center during Yingmin Li's summer internship there. We also wish to thank Michael Huang and the reviewers for their helpful comments on this work.

## References

- [1] A. Bakker and J. Huijsing. *High-Accuracy CMOS Smart Temperature Sensors*. Kluwer Academic, Boston, 2000.

- [2] L. A. Barroso, J. Dean, and U. Hölzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, April 2003.
- [3] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6), 2003.
- [4] J. Burns and J.-L. Gaudiot. Area and system clock effects on SMT/CMP processors. In *Proc. PACT 2001*, pages 211–18, Sep. 2001.
- [5] J. Clabes et al. Design and implementation of the power5 microprocessor. In *ISSCC Digest of Technical Papers*, pages 56–57, Feb. 2004.
- [6] J. Donald and M. Martonosi. Temperature-aware design issues for smt and cmp architectures. In *Proceedings of the 2004 Workshop on Complexity-Effective Design*, June 2004.
- [7] L. Hammand, B. A. Nayfeh, and K. Olukotun. A single-chip multiprocessor. *IEEE Computer*, 30(9):79–85, Sep. 1997.
- [8] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *Proc. ISLPED'03*, Aug. 2003.
- [9] Z. Hu, D. Brooks, V. Zyuban, and P. Bose. Microarchitecture-level power-performance simulators: Modeling, validation, and impact on design, Dec. 2003. Tutorial at Micro-36.
- [10] W. Huang, M. R. Stan, K. Skadron, S. Ghosh, K. Sankaranarayanan, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proceedings of the 41st Design Automation Conference*, June 2004.
- [11] W. Huang, M. R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghosh, and S. Velusamy. Compact thermal modeling for temperature-aware design. Technical Report CS-2004-13, Univ. of Virginia Dept. of Computer Science, Apr. 2004.
- [12] R. Kalla, B. Sinharoy, and J. Tandler. Power5: Ibm's next generation power microprocessor. In *Proc. 15th Hot Chips Symp*, pages 292–303, August 2003.
- [13] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. Comparing Power Consumption of SMT DSPs and CMP DSPs for Mobile Phone Workloads. In *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2001)*, Nov. 2001.
- [14] K. Krewell. UltraSPARC IV mirrors predecessor: Sun builds dual-core chip in 130nm. *Microprocessor Report*, pages 1, 5–6, Nov. 2003.
- [15] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In *Proc. Micro-36*, pages 81–92, Dec. 2003.
- [16] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In *Proc. ISCA-31*, pages 64–75, June 2004.
- [17] Y. Li, D. Brooks, Z. Hu, K. Skadron, and P. Bose. Understanding the energy efficiency of simultaneous multithreading. In *Proc. ISLPED'04*, Aug. 2004.
- [18] D. T. Marr, F. Binns, D. L. Hill, G. Hinton, D. A. Koufaty, J. A. Miller, and M. Upton. Hyper-threading technology architecture and microarchitecture. *Intel Technology Journal*, 6(1):4–15, Feb. 2002.
- [19] M. Moudgill, J.-D. Wellman, and J. H. Moreno. Environment for powerpc microarchitecture exploration. *IEEE Micro*, 19(3):15–25, 1999.
- [20] R. Sasanka, S. V. Adve, Y. K. Chen, and E. Debes. The energy efficiency of cmp vs. smt for multimedia workloads. In *Proc. 18th ICS*, June 2004.
- [21] J. Seng, D. Tullsen, and G. Cai. Power-sensitive multithreaded architecture. In *Proc. ICCD 2000*, pages 199–208, 2000.
- [22] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proc. ASPLOS-X*, Oct. 2002.
- [23] SIA. *International Technology Roadmap for Semiconductors*, 2001.
- [24] K. Skadron. Hybrid architectural dynamic thermal management. In *Proc. DATE'04*, Feb. 2004.
- [25] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan. Temperature-aware microarchitecture. In *Proc. ISCA-30*, pages 2–13, Apr. 2003.
- [26] A. Snively and L. Carter. Multi-processor performance on the tera mta. In *Proc. 12th ICS*, May 1998.
- [27] D. M. Tullsen, S. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proc. ISCA-22*, 1995.
- [28] V. Zyuban and P. Strenski. Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit levels. In *Proc. of ISLPED'02*, August 2002.