The Energy Efficiency of CMP vs. SMT for Multimedia Workloads*

Ruchira Sasanka Sarita V. Adve
Department of Computer Science
University of Illinois at Urbana-Champaign
{sasanka, sadve}@cs.uiuc.edu

Yen-Kuang Chen Eric Debes
Architecture Research Lab
Intel Corporation
{yen-kuang.chen, eric.debes}@intel.com

Abstract

This paper compares the energy efficiency of chip multiprocessing (CMP) and simultaneous multithreading (SMT) on modern out-of-order processors for the increasingly important multimedia applications. Since performance is an important metric for real-time multimedia applications, we compare configurations at **equal performance**. *We perform this comparison for a large number of performance points derived using different processor architectures and frequencies/voltages.*

We find that for the design space explored, for each workload, at each performance point, CMP is more energy efficient than SMT. The difference is small for two thread systems, but large (18% to 44%) for four thread systems. We also find that the best SMT and the best CMP configuration for a given performance target have different architecture and frequency/voltage. Therefore, their relative energy efficiency depends on a subtle interplay between various factors such as capacitance, voltage, IPC, frequency, and the level of clock gating, as well as workload features. We perform a detailed analysis considering these factors and develop a mathematical model to explain these results.

Although CMP shows a clear energy advantage for four-thread (and higher) workloads, it comes at the cost of increased silicon area. We therefore investigate a hybrid solution where a CMP is built out of SMT cores, and find it to be an effective compromise. Finally, we find that we can reduce energy further for CMP with a straightforward application of previously proposed techniques of adaptive architectures and dynamic voltage/frequency scaling.

Categories and Subject Descriptors

C.1.2 [**Processor Architectures**]: Multiple Data Stream Architectures

General Terms

Performance, Design

Keywords

CMP, SMT, Energy Efficiency, Multimedia

*This work is supported in part by an equipment donation from AMD Corp., a gift from Intel Corp., and the National Science Foundation under Grant No. EIA-0103645, CCR-0209198, CCR-0205638, EIA-0224453, and CCR-0313286. Sarita V. Adve was also supported by an Alfred P. Sloan Research Fellowship. Ruchira Sasanka was supported by an Intel graduate fellowship and began this work as a summer intern at Intel.

ICS'04, June 26–July 1, 2004, Saint-Malo, France.

1. Introduction

This paper compares the energy efficiency of chip multiprocessing (CMP) [10] and simultaneous multithreading (SMT) [19] for multimedia applications on modern out-of-order general-purpose processors (GPPs). Multimedia applications are becoming increasingly important for GPPs in a variety of systems including desktops, laptops, tablet PCs, and likely future handheld devices. GPPs have begun to support multithreading for improved throughput, using either CMP or SMT. These techniques are a good match for multimedia applications which are inherently multithreaded. However, multimedia applications often run on portable systems facing strict energy constraints. It is therefore important to study the energy efficiency of general-purpose CMP and SMT architectures for multimedia applications.

SMT allows multiple application threads to be run at the same time, within the same processor, potentially increasing utilization of the processor resources. Specifically, current wide issue out-of-order processors are often unable to utilize the full supported fetch/decode/issue width for a single thread. SMT utilizes these otherwise wasted resources for other threads, potentially improving total throughput with little additional hardware. CMP, on the other hand, improves throughput by adding additional processors rather than improving their utilization.

At first glance, SMT may appear to be inherently more energy efficient than CMP since it potentially uses its resources more effectively – SMT can get more IPC (instructions per cycle) from less hardware. However, in reality, the comparison is more complex, both in the analysis to understand the experimental results and in the methodology to generate the right results.

Sources of complexity and our solutions. For real-time multimedia applications, performance is a key constraint. A fair comparison of energy must therefore also consider performance. As a result, we compare the energy of SMT and CMP *at the same performance*, and perform this comparison for a wide range of performance points. The complexity arises because each performance point can be obtained by CMP and SMT using several combinations of frequency and processor microarchitecture (referred to as *the core architecture*). For example, a narrow width core architecture at a high frequency or a wider width core architecture at a lower frequency can achieve the same performance but at different energy. A fair comparison must consider the combinations that provide the minimum energy for SMT and CMP at that performance point. This best combination for SMT could be different from that for CMP, both could differ for different workloads and different performance points, and both are difficult to determine a priori. Given that at the fairest point of comparison, the processor core architectures and frequencies employed by CMP and SMT may be different and are not known a priori, it is no longer clear which technique is most energy efficient.

The complexity of the problem becomes most evident when we try to analyze our results. Multiple subtle interactions dictate which configuration would be most energy efficient for either CMP or SMT, and which of these two best configurations provides the lowest overall energy. The analysis involves consideration of the amount of clock gating employed, the increase in $Power/IPC^x$ (generally $2 \leq x \leq 3$) with increasing processor core complexity, and properties of the workload that make them amenable to performance speedups from CMP and SMT.

Methodologically, the complexity arises because, as implied by the above discussion, we must identify and explore a large design space, and carefully choose the specific pairs of SMT and CMP configurations that should be compared against each other. To bound the design space explored, this work focuses on out-of-order superscalar processors, based on contemporary general-purpose designs. Undoubtedly, there are other architectures that could be more energy efficient for these applications; however, we chose out-of-order superscalar processors since our focus is on general purpose processors which are becoming increasingly important for portable devices (e.g., Pentium-M). We simulate processor core complexities ranging from 2-wide to 8-wide fetch/decode width. To consider a full performance continuum, we evaluate the considered processor core architectures over a range of frequencies, from 600 MHz to 1.6 GHz (with corresponding voltages). For a given workload, we compare the energy efficiency of CMP and SMT by considering configurations that provide *the same performance*, and perform such comparisons for all performance points in the investigated design space. For each performance point, we identify the configuration (i.e., core architecture and frequency) that gives the minimum energy for SMT and CMP, and compare these minimum energy values.

We consider two- and four-thread workloads derived from combinations of 8 single-threaded multimedia benchmarks consisting of low and high bit rate video and speech codecs. (N-thread workloads run on an N-thread SMT or on an N-core CMP.)

Findings. Although SMT is known for its efficiency in utilizing resources, we find that CMP is consistently more energy efficient than SMT (comparable with two threads and significantly better with four threads). More specifically, our results show that for the design space explored, for each workload, at each performance point, (1) the least-energy CMP configuration showed lower energy than the least-energy SMT configuration, (2) the energy difference was larger at the high-performance points and for four-thread workloads (for four-thread workloads, the average benefit of CMP over SMT was 44% for the highest performance points and 18% for the lowest performance points), and (3) the least-energy SMT configuration had moderately higher complexity and higher frequency/ voltage. To understand the reasons for our results and to extend our findings to other systems and workloads not simulated here, we perform a qualitative analysis and develop an analytic model that exposes a subtle interplay between various factors affecting the results.

Broader implications. Our results have two broad implications beyond simply a comparison of SMT and CMP. First, our results clearly underscore the advantage of CMP for four-thread (and higher) workloads for our applications. However, a four-core CMP will have a much larger silicon area than an SMT with moderately higher core complexity. Thus, the energy advantage of CMP comes at an area cost. To get the best of both worlds, for four-thread workloads, we study a hybrid CMP/SMT architecture **(HYB)** where a CMP is built out of SMT cores (e.g., IBM Power5). We find such a two-core CMP with two-thread SMT cores has significantly higher energy efficiency than a pure SMT processor. The hybrid architecture with two SMT cores generally needs less silicon area than a 4-core CMP. Moreover, such an architecture will also provide better energy for workloads that would not scale well in performance across a four core CMP.

The second broad implication arises from our observation that although CMP configurations generally give the best energy efficiency, different configurations are optimal at different performance points (as is the case with SMT). This motivates the use of recently proposed adaptive architecture and frequency/voltage scaling techniques. We also find that applying these techniques independently on the different CMP processor cores to create a heterogeneous CMP provides even further energy savings for CMP. While it may be possible to apply some of these techniques to SMT, it is as yet unclear how this can be done.

2. Related Work

Although there is significant prior work comparing the performance of CMP and SMT [20, 10], comparing the energy efficiency of SMT with a superscalar [17], and comparing the area and layout overheads of SMT and CMP [6], there is very little prior work on energy related comparisons of SMT and CMP. The only such work, to our knowledge, is by Kaxiras et al., also in the context of multimedia applications [14]. However, that work considers a VLIW processor core (which produces low IPC for the compiled codes studied) and primarily compares average power at a given frequency for CMP vs. SMT. It examines only two alternative core architectures and only one workload. In contrast, we study out-of-order superscalar processors (which give higher IPCs) and compare energy at the same performance, for a wider range of workloads and core processor architectures. The primary conclusion from the work by Kaxiras et al. is that at equal frequency, CMP consumes more power than SMT. This does not contradict our results since we also find that CMP configurations have higher maximum and average power than corresponding SMT configurations at a fixed (highest) frequency (e.g., leftmost points of Figure 3(a) and (b)). A more detailed comparison of the two studies appears in the extended version of this paper [15].

3. Experimental Methodology

3.1 Systems Modeled

3.1.1 Design Space & Naming Convention

We model two **classes of systems** - one supporting two threads and the other supporting four threads as illustrated in Figure 1. For the two-thread systems, we model a single core SMT that supports two threads and a 2-core CMP (each core supports one thread). For the four-thread systems, we model a single core SMT that supports four threads, a 4-core CMP (each core supports one thread), and a hybrid system, which is a 2-core CMP with each core being an SMT supporting two threads.

To bound the design space explored, we focus on out-of-order processors. To adequately represent this design space, we model several *core architectures* for the out-of-order processor cores, ranging from a fetch/decode width of two to eight. For each fetch/decode width, we appropriately scale other resources (e.g., instruction window size and the number of functional units), as discussed in Section 3.1.2. For a given CMP, we assume all processor cores have the same configuration (except when we consider adaptive architectures in Section 4.4.2).

We adopt the following naming convention, as also shown in Figure 1. For a two thread system, we denote an SMT or CMP system with cores with a fetch/decode width of N by **SMT2-**N

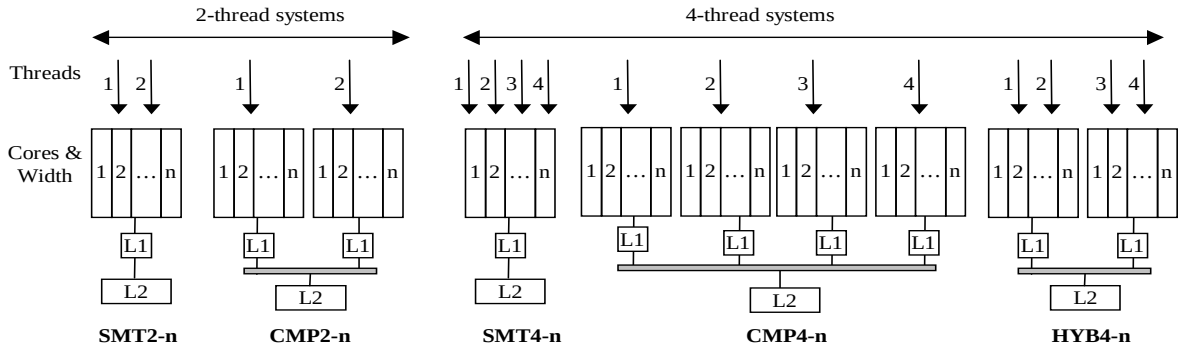


Figure 1. **Systems modeled and naming convention. An SMT, CMP or HYB processor is named as NAME<# of threads>–<fetch/decode width of a core>, where NAME is SMT, CMP or HYB.**

and **CMP2-**N respectively. Note that CMP2-N has two N-wide cores whereas the SMT2-N has only one N-wide core. Similarly, we use **SMT4-**N, **CMP4-**N, and **HYB4-**N to denote four-thread SMT, CMP, and HYB systems (respectively) with cores that have a fetch/decode width of N.

3.1.2 Processor Core & Memory Parameters

Table 1 summarizes the processor core and memory hierarchy parameters used for the experiments reported here – *Width* refers to the fetch/decode width and *Threads* is the number of threads supported by a processor core. Given that there are many ways to reasonably scale processor core parameters with fetch/decode width and many ways to set memory hierarchy parameters, we explored a few alternate parameterizations (e.g., Instruction window size = $Width$ x 16) and found that the trends were the same as those presented here.

In all cases, the processor core is an out-of-order superscalar, modeled after the MIPS R10000 superscalar core. Specifically, the register file is separate from the reorder buffer as in modern implementations. For SMT, we assume that all concurrent threads share most resources in the processor, including the instruction window, functional units, L1 caches, and register files; however, each thread is given a separate branch prediction table and a return address stack. Further, 32 additional integer and floating point registers are assumed for each additional thread, to capture the architectural state. Sizes of the instruction TLB and data TLB are also increased to support additional threads on SMT processors as given in Table 1. We use the ICOUNT policy [19] to prioritize instruction fetch from different threads.

We model L1 instruction and data caches and a unified L2 cache. All threads of an SMT share the L1 data and instruction caches. For CMP, each processor core has its own L1 data and L1 instruction cache, and all cores share an L2 cache through a common bus. The media applications studied have a relatively small working set and a high computation to memory ratio. Therefore, relatively smaller L1 caches are sufficient to obtain very high hit rates. Both L1 instruction and data cache sizes were selected after a sensitivity analysis. For each application, we determined the minimum cache size necessary to obtain a hit ratio over or close to 99% for the data cache and a hit ratio over or close to 98% for the instruction cache. Across all applications, the largest such size was found to be 8K for data and 16K for instruction cache. Therefore, as summarized in Table 1, each CMP processor core was given an 8K L1 data cache and a 16K L1 I cache for all CMP systems. SMT cores were given the same amount of cache per thread supported (e.g., two-thread SMT has a 16K L1 data and 32K L1 instruction cache). Consequently, in a given system, both CMP and SMT processors have the same total amount of cache. (We also evaluated another set of cache parameters where the SMT cache size was smaller than the combined cache size on the corresponding CMP; e.g., for 4-thread systems, we used 16K data cache for SMT and 8K for each of the four CMP cores for a total of 32K. The overall trends in the results were the same as those reported here.) All caches are non-blocking and writeback. A relatively large cache line size of 64B was found to be beneficial for these applications due to their streaming nature.

We did not perform a detailed sensitivity analysis for the L2 cache because (a) it does not have much impact on performance since L1 hit ratios are very high, (b) the average power for the L2 cache is a small fraction of the total even for the large 1MB L2 cache we model (see Figure 5), and (c) the L2 cache is common to all systems. For the same reasons, although we measure the energy of the L2 cache, we do not include it in the total energy reported in our comparisons.

We investigate all the core architectures at frequencies ranging from 600MHz to 1.6GHz with voltages scaled according to data for the Intel Pentium-M (Centrino) mobile processor which supports this frequency range [1]. The configurations of a core architecture at different frequencies may be interpreted as a single processor supporting dynamic frequency and voltage scaling (DVS) or as different fixed-frequency processor designs. Note that, compared to wider processors, it may be possible for narrower processors to support a higher frequency at a given voltage. We did not model this effect since it is difficult to do so accurately and would only further favor CMP processors, as easily confirmed by our analysis in Section 4.3.

Henceforth, we use the term ***core architecture*** to refer to the fetch/decode width and other parts dependent on this width. We use the term ***configuration*** to refer to a combination of the core architecture and frequency. For a given number of threads t and core architecture c, we refer to CMPt-c (or SMTt-c or HYBt-c) as the ***system architecture*** and the combination of system architecture and frequency as the ***system configuration***.

3.2 Workloads

We consider eight single-thread multimedia benchmarks covering high and low bit rate video and speech codecs. These benchmarks are summarized in Table 2(a) and described in more detail in previous work [11]. They form the core components of many high-level multimedia applications (e.g., video teleconferencing, DVD playback, video editing) and are representative of most widely used media benchmarks. A real system would run several of these benchmarks together as part of one or more such high-level applications. For example, a video teleconferencing application between N participants at N different sites would involve one video and speech encoder and N-1 video and speech decoders at each site. A participant may receive high or low bit rate streams depending on the computation power and bandwidth available at the other participating sites; therefore, a site may need to support different types of decoders and encoders within the same application. Moreover, even a given benchmark may be parallelized creating

Width Dependent Parameters	
Fetch/decode rate	*Width* $\in \{2, 3, 4, 5, 6, 7, 8\}$
Instruction window (reorder buffer) size	24 * *Width*
# of integer functional units	*Width*
# of floating point units	1 (*Width* $\leq$ 3), 2 (*Width* $\leq$ 6), 3 (*Width* $>$ 6)
# L1 ports	2 (*Width* $\leq$ 6), 3 (*Width* $>$ 6)
# Address generation units	2 (*Width* $\leq$ 6), 3 (*Width* $>$ 6)
Retirement rate	# Int units + # FP units + # L1 ports
Load/Store buffer size	8 * *Width*
Thread Dependent Parameters	
L1 data cache	8K * *Threads*, Write Back
L1 data cache associativity	4 (*Threads* $\leq$ 2), else 8
L1 instruction cache	16K * *Threads*, 4-way
iTLB & dTLB size	128 entries * *Threads*
Branch prediction	2K entries * *Threads*, bimodal agree
Return address stack size	16 * *Threads*
Integer register file size	32**Threads* + reorder buffer size
Float register file size	32**Threads* + (reorder buffer size)/2

Common Processor Parameters	
Processor core speed	600 MHz to 1.6 GHz (voltage scaled)
Process technology	0.13 micron
Integer FU latencies	1/4/12 add/mult/div (pipelined)
FP FU latencies	4 default, 12 div. (all but div. pipelined)
Branch penalty	16 cycles
Bit widths	64-bit data / 48-bit address
Common Memory Hierarchy Parameters	
L1 & L2 cache line size	64B
L2 cache (on chip)	1MB, 16-way associative, write back 64B line, 1 port
Main Memory	16B/cycle, 4-way interleaved
Common Contentionless Memory Latencies	
L1 instruction cache access time	1 cycle
L1 data cache access time	2 ($\leq$ 16K), 3 (32K)
CMP bus latency	2 cycles
L2 cache access time (on chip)	8 cycles
Main Memory	100 cycles (L2 miss to memory)

Table 1. Processor core and memory parameters. $Width$ refers to the fetch/decode width of a core and $Threads$ refers to the number of threads supported by a single core.

Benchmark	Type of codec	Input size (Frames)	Base IPC
GSMd	Speech	1000	3.4
GSMe		1000	3.5
G728d	Speech	1000	2.2
G728e		1000	1.9
H263d	Video	450	3.1
H263e		50	2.0
MPGd	Video	200	2.8
MPGe		50	1.5

(a)

2-thread workload	Total IPC	2-thread workload	Total IPC
MPGe_MPGe	2.9	MPGe_GSMd	4.9
MPGe_G728d	3.7	H263e_H263d	5.0
H263e_H263e	3.9	H263d_G728d	5.3
G728e_G728d	4.1	H263e_GSMe	5.4
MPGe_MPGd	4.2	GSMe_G728d	5.7
H263e_MPGd	4.7	H263d_GSMe	6.6
H263d_G728e	4.8	GSMe_GSMe	7.0

(b)

4-thread workload	Tot. IPC
MPGe_MPGe_MPGe_MPGe	5.8
MPGe_MPGe_G728e_G728d	7.0
H263d_G728e_H263d_G728e	9.6
H263e_H263d_H263e_GSMe	10.4
MPGe_MPGd_GSMe_GSMe	11.2
MPGd_GSMd_H263d_G728d	11.4
H263d_GSMe_H263d_GSMe	13.2
GSMe_GSMe_GSMe_GSMe	13.9

(c)

Table 2. (a) Single-thread benchmarks with IPC on most aggressive processor, (b) two-thread workloads, and (c) four-thread workloads. (b) and (c) are ordered by the sum of the IPCs of the constituent threads when run individually on the most aggressive core configuration.

multiple independent threads that process different frames independently. Thus, we can envisage realistic workloads consisting of a number of different copies of different combinations of the benchmarks in Table 2(a) with each copy working on its own data.

For the small-scale systems studied here (two or four thread CMP and SMT), we can assume that the total number of threads available for running in a realistic system will be larger than the number of simultaneous threads supported in the system. A real-time operating system (RTOS) must therefore choose which combination of threads to co-schedule at each time, with consideration for any synchronization among related threads (e.g., audio and video for the same stream). The co-scheduling algorithm can have an impact on the overall performance, but real-time co-scheduling algorithms for SMT are still an open area of research [13]. To eliminate dependence on the co-scheduling algorithm, we report results separately for different combinations of N threads for an N-thread system. Thus, for a two-thread system, we separately report results for different pairs of the eight benchmarks of Table 2(a). The actual performance of a full real-time application would depend on how often the RTOS co-schedules each of the specific combinations for its chosen co-scheduling policy.

Furthermore, a real RTOS scheduling policy may co-schedule different parts of N concurrent benchmarks at different times. For example, consider two benchmarks with very different execution times per frame. The shorter frame may be co-scheduled along with any part of a longer frame and it is possible the execution characteristics are different depending on when the two frames are co-scheduled. Again, to report results independent of the co-scheduling policy and to average out such differences, we run several frames of the co-scheduled benchmarks to get the average behavior for that benchmark combination. The maximum number of frames we consider for each benchmark is reported in Table 2(a) along with the IPC on an 8-wide (superscalar) core for each application.

Studying all combinations of two and four out of our eight benchmarks would have resulted in an inordinately large number of workloads (e.g., 36 possibilities for 2-thread systems). We therefore selected a representative subset of these, summarized in Table 2(b) and (c), using a methodology described further in [15]. The four-thread workloads are combinations of the two-thread workloads [15]. When running on the HYB system, the constituent two-thread pairs are paired again for each SMT core.

For workloads with threads from different applications, there is an inherent problem when attempting to compare the same amount of work on all system architectures. For the first application of a given workload, we run the same number of frames for all CMP and SMT system architectures. During that time, the amount of work done by the other threads can change slightly for different system architectures. This is an inherent property of SMT and CMP architectures. Since we assume that the RTOS has enough threads to schedule and we want to maximize the throughput of the system, we run the other threads until the first thread finishes its maximum number of frames. Thus, the first thread executes the same number of instructions on all core architectures. To overcome the problem of the other threads executing a slightly different number of instructions on different system architectures, when

comparing energy efficiency, we use energy and execution time metrics that are normalized to the number of instructions executed – Energy Per Instruction (EPI) and Time Per Instruction (TPI).[1] Note that this problem does not arise when all threads in the workload are from the same application, since all threads finish almost at the same time due to the fairness of SMT's ICOUNT policy and the symmetry of CMP. We study several such workloads for all systems, and they follow the same overall patterns as the others. We also measured the discrepancy in instruction counts and found that it was $< 5\%$ for most and $< 12\%$ for all workloads.[2]

3.3 Simulation Environment and Methodology

We model the performance of the systems in Section 3.1 using a version of the RSIM simulator [12] modified to support both SMT and CMP. RSIM is an execution-driven, cycle level simulator that models the full impact of branch and address speculation (e.g., modeling wrong path instructions) and contention at all resources. All applications are compiled with the SPARC SC4.2 compiler with full optimization (O4). Previous work showed that for the benchmarks studied here, performance scales virtually linearly with processor frequency, since the amount of time spent on memory stalls is negligible [11]. We therefore run simulations at one base frequency, and use linear scaling to obtain execution time at other frequencies. We validated this by running actual simulations with frequencies at 100MHz intervals. (It was impractical to simulate all frequencies over the 1GHz range reported here.)

We use a combination of tools integrated with RSIM to model the dynamic and static energy of all systems. To model dynamic energy of processor cores and caches, we use the Wattch tool [5] integrated with RSIM. Wattch is enhanced to model duplicated resources, additional tags for thread IDs, etc. For CMP, we model the energy consumption of the bus between the L1 and L2 caches using models from the Orion project [21].[3] We assume a bus length of 5mm with two cores and 10mm with four cores. However, as shown in Figure 5, the bus energy is very small. We also model the static energy consumption of major structures like caches, register files, and the instruction window using the HotLeakage model [22]. For this purpose, we model the temperature of major structures on the processor using temperature models described in [18]. However, for the structures we model and for the 0.13 micron technology parameters we use, we find that the leakage power is less than 2% of the dynamic power.

We assume aggressive clock gating for all CMP and SMT cores, as in many current general-purpose processors [3]. Although it is relatively easy to disable clocking of unused ports of the multi-ported structures and unused functional units, it is practically not possible to achieve 100% clock gating. Some clock gating events are expensive to identify and some of the gating is foregone to avoid lengthening critical paths, race conditions, unequal clock distributions, extensive validation and di/dt effects [9]. We assume that all but 10% of the unused circuitry can be turned off with clock gating in typical industrial clock-gated circuits, as mentioned by Brooks et al. [5]. (For reference, for the Pentium 4 processor, the idle power consumption was found to be 15% to 20% of the power consumed when running MPEG [7].) We later discuss the sensitivity to this parameter and give results with 20% ungated circuitry and with no gating at all (Section *4.3.1*).

[1] We cannot use IPC since we vary the frequency as well. $TPI = 1/(\text{Frequency} \times \text{IPC})$.

[2] The only other reasonable alternative would be to consider a specific number of frames from the other threads. This has the drawback that some thread contexts/cores will remain idle when the shorter thread(s) finishes, and would likely favor CMP (since the idle processor could be deactivated with CMP). Nevertheless, we evaluated several workloads with this alternative and found the results to be similar.

[3] We thank Li-Shiuan Peh and Hang-Sheng Wang for quickly generating and providing us the bus models.

3.4 Metrics and Representation of Collected Data

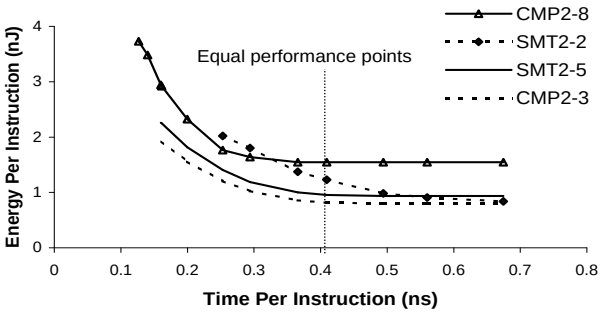


Figure 2. Example EPI vs. TPI graph.

When considering energy as a metric of comparison, one must also consider the performance obtained. One common metric used is the energy-delay product (i.e., energy/performance) [8]. However, this metric is unsatisfactory if the user desires a fixed amount of performance or is constrained by a fixed amount of energy (e.g., battery life). Specifically, for real-time applications such as those studied here, there is often a fixed desirable performance target (derived from the application deadlines and the rest of the load on the system). We therefore focus our effort here on understanding optimal energy configurations, given a fixed performance target (the data and analysis for the optimal performance configuration for a fixed energy target is similar). We report results for the energy-delay product and other more conventional metrics in [15]. Recall from Section 3.2 that we use the normalized energy per instruction (EPI) and time per instruction (TPI) to measure energy and performance respectively.

For a CMP or SMT with a given core architecture, varying the processor frequency provides a continuum of performance points. Any of these points could be achieved either as a fixed-frequency design or in a system with DVS support. We collect data for CMP and SMT systems using all combinations of core architectures and frequencies given in Section 3.1. For a given class of systems (2 or 4 thread), for each workload, for each performance point (i.e., TPI), we compare all the system configurations that provide that performance, to determine which system gives the least energy for that performance. In general, the lowest EPI system is different for different performance points.

Figure 2 illustrates how we represent the collected data to perform this comparison for 2-thread systems. It plots EPI versus TPI for the MPGe_MPGd workload. Each curve in the figure represents one core architecture for an SMT or CMP system and each point on a given curve represents a different frequency (from 1.6GHz on the left to 600MHz on the right). Only four systems are shown for clarity - CMP2-8, SMT2-2, SMT2-5 and CMP2-3. The points along a vertical line on this graph represent points of equal performance. The lowest point on the line represents the configuration that gives the least energy for that performance. For example, for a target TPI of 0.4ns, CMP2-3 provides the least energy. CMP2-3 also turns out to provide the least-energy for a large performance range for this workload.

In general, the least-energy system architecture may be different in different performance ranges for two reasons. First, not all system architectures may be able to provide all performance points on the extreme left and right sides of the TPI axis. For example,

in Figure 2, CMP2-8 is least-energy for a few of the leftmost TPI values, because CMP2-3 is unable to provide that level of performance even at the maximum frequency. Second, in the middle performance ranges, the least-energy configuration could change if the curves of two architectures cross. This crossing can occur due to the non-linearities in the voltage vs. frequency curve.

4. Results

We performed all our analysis using graphs such as shown in Figure 2 – one graph per 2-thread or 4-thread workload, 14 curves per 2-thread graph (7 each for CMP and SMT), and 21 curves per 4-thread graph (7 each for CMP, SMT, and HYB). To distill the information from these graphs into a more readable form, Figures 3(a) and (b) show the EPI values of the *best* SMT and the *best* CMP architectures for the entire performance range. Figure 3(b) shows the *best* HYB architecture as well. To save space, only a few representative workloads are shown since other workloads follow similar trends. Since one system architecture is the best-energy architecture for a range of performance points (using different frequencies), we label the performance points at which the best system architecture changes (going from left to right). The best architectures are marked as sn, cn, and hn to indicate SMT, CMP, HYB respectively, with a core fetch/decode width of n.

Tables 3(a) and (b) supplement the above graphs by tabulating the magnitude of the EPI difference between the best SMT and CMP configurations, as a percentage of the SMT configuration. Since we cannot tabulate each of the infinite performance points and since an average over the entire space is not too meaningful, we divide the TPI axis on the EPI-TPI time graphs into three regions (high, medium, and low), based on the performance degradation relative to the highest performance configuration (which is always the 8-wide, 1.6GHz CMP for all workloads). For two-thread workloads, the performance degradation is from 1X to 1.5X for the first region (highest performance), 1.5X to 3X for the second region (medium performance), and $>$ 3X for the third region (lowest performance). For four-thread workloads, the performance degradation is from 1X to 2X, 2X to 4X, and $>$ 4X respectively for the three regions. For the highest performance region, we only include points where at least one SMT configuration can achieve that performance.

For each region, the tables give the average percentage improvement (in EPI) of the best CMP configuration over the best SMT configuration (and best hybrid configuration for four-threads) at different performance points in this region. The average is calculated by finding the area between the two curves for that region and dividing that by the TPI difference for that region. Note that for a given region in this table, the optimal CMP, SMT, and HYB architectures may be different at different points in the region.

The EPI vs. TPI graphs as shown in Figure 3(a) and (b) also convey information regarding *absolute performance* and *average power*. The performance of systems with 8-wide cores at 1.6GHz is given by the topmost points on each curve. Further, for a given TPI, EPI is proportional to the *average power*. Since the discussion below focuses on energy at equal performance (i.e., EPI for given TPI), it follows that the discussion also applies to average power (for different performance points).

4.1 Results Across All Configurations

Our data shows that for all our systems and workloads, for all performance regions, a CMP architecture gives the least EPI.

Workload	High	Med	Low
MPGe_G728d	4	3	2
H263d_G728e	5	4	1
MPGe_MPGe	7	7	2
G728e_G728d	8	6	3
H263e_H263e	8	6	4
H263e_H263d	8	8	4
MPGe_MPGd	9	6	3
H263e_MPGd	10	8	3
H263e_GSMe	10	10	4
H263d_G728d	10	10	4
GSMe_G728d	11	10	5
MPGe_GSMd	13	10	6
H263d_GSMe	15	16	8
GSMe_GSMe	19	18	9
Average	10	9	4

(a)

Workload	CMP vs. SMT			CMP vs. HYB		
	Hi	Med	Lo	Hi	Med	Lo
MPGe_MPGe_MPGe_MPGe	30	23	10	10	7	5
MPGe_MPGe_G728e_G728d	30	24	10	10	8	5
H263d_G728e_H263d_G728e	42	37	16	6	5	1
H263e_H263d_H263e_GSMe	45	39	19	9	10	6
MPGe_MPGd_GSMe_GSMe	47	43	21	18	15	8
MPGd_GSMd_H263d_G728d	50	46	22	17	15	7
H263d_GSMe_H263d_GSMe	53	51	25	13	14	10
GSMe_GSMe_GSMe_GSMe	54	49	24	21	17	10
Average	44	39	18	13	11	6

(b)

Table 3. Range of % EPI savings of the best-energy CMP over the best-energy SMT and HYB for different performance regions (a) two-thread workloads and (b) with four-thread workloads for high, medium, and low performance regions.

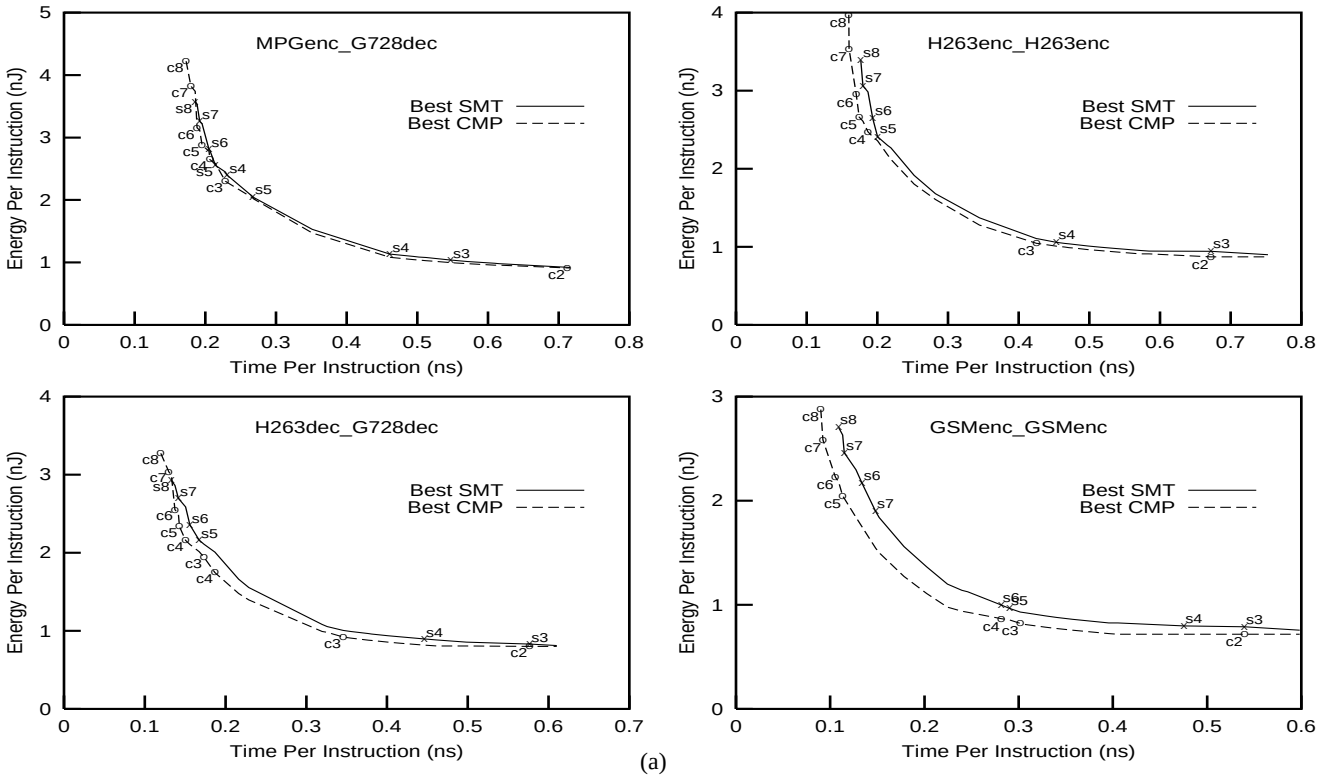
Comparing CMP and SMT, for two thread workloads, the difference between them is mostly small (average 10%, 9%, and 4% respectively over the three regions). For four thread workloads, CMP is significantly better than SMT (average 44%, 39%, and 18% for the three regions). In both cases, the difference increases with increasing performance.

Focusing on HYB, our data shows that it is significantly more energy efficient than SMT and comes close to CMP for many performance points and workloads. On average, the difference between CMP and HYB is reduced to 13%, 11%, and 6% for the three regions, respectively.

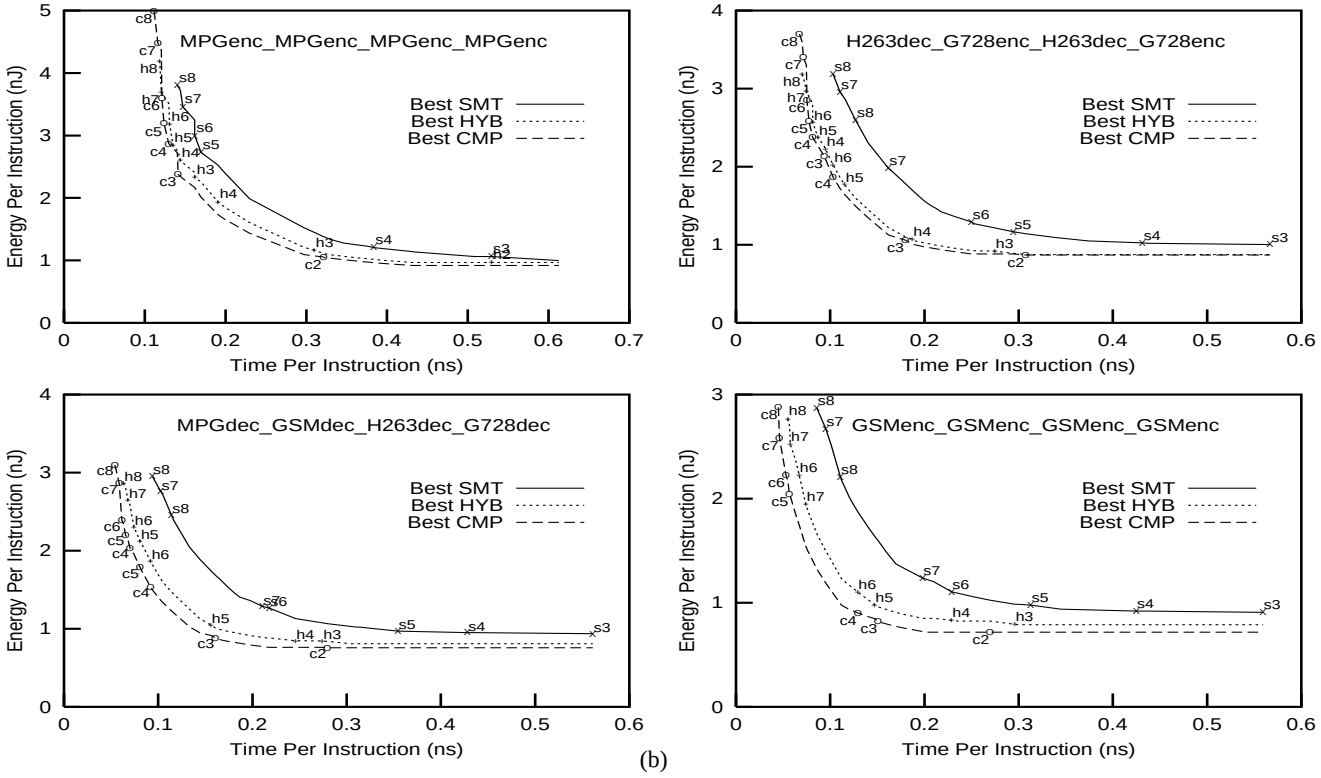
4.2 The Best Core Architectures

Figures 3(a) and (b) show that for all the workloads, the best CMP uses a less or equally complex core architecture (i.e., lower or same fetch/decode width) than the best SMT and the best HYB at any given performance point. HYB is closer to CMP than SMT. (The total resources available to CMP, however, are larger because CMP has more processor cores.)

Further, different core architectures are the best in different performance regions. Systems with fixed (vs. adaptive) architectures need to pick one *overall best core architecture* to implement. We define this to be the architecture that, when averaged across all performance points, has the least EPI difference from the best core at the same performance point. Note, however, that the overall best architecture may not have performance points for the entire performance region covered by all core architectures. Further, this also assumes the presence of dynamic frequency/voltage scaling since



(a)



(b)

Figure 3. **EPI for best-energy SMT, best-energy CMP and, best-energy HYB configuration at different performance points for (a) two-thread workloads and (b) four-thread workloads.** The best system configurations are marked as sn, cn, or hn to represent SMT, CMP, and HYB, respectively, with cores of fetch/decode width of n. They are marked only at the points where the best configuration changes, going from left to right.

an architecture is best only at the appropriate frequency. We find a 4-wide core to be the overall best for CMP for both two and four thread systems, a 5-wide core is best for SMT2 and HYB4, and a 6-wide core is best for SMT4. More details are in [15].

4.3 Analysis of the Results

Section 4.3.1 provides a qualitative analysis for the underlying reasons for the superior energy efficiency of CMP over SMT, and determines factors that could make SMT more energy efficient. Section 4.3.2 formalizes the intuition from Section 4.3.1 by providing a mathematical model. These sections are complex, but essential to understand our results and extrapolate to other system parameters and workloads.

4.3.1 Qualitative Understanding

We know that $EPI = Power \times TPI = \alpha C V^2 f \times TPI$, where α is the switching activity factor, C is the total capacitance, V is the supply voltage, and f is the frequency.[4] α and C depend on the core architecture. We refer to the reciprocal of $\alpha C V^2 f$ as the *energy efficiency*, or simply the *efficiency*, of the corresponding system configuration. It follows that for a given class of system (superscalar, CMPN, or SMTN, where N is the number of threads) and a given TPI, the minimum EPI occurs for the configuration that has the highest energy efficiency as defined above.

Figure 4 illustrates the effect of each factor in the energy efficiency term. *For a specific TPI* (selected from the middle part of the performance range), for CMP2 and SMT2 running the workload MPGe_MPGe, parts (a)-(f) of the figure respectively show the variation with core complexity (i.e., fetch/decode width) of IPC, C, α, αC, $V^2 f$, and EPI. Note that for the $V^2 f$ plot, the frequency value for a given core width is determined as the ratio of TPI and IPC with that core. We show the graph of the product αC due to its significance – αC is proportional to the average power at a given voltage/frequency. It is also interesting to see the contribution to average power from different structures. Figure 5 shows this information for four systems and the MPGe_MPGe workload.

To understand when the lowest EPI is obtained, the following discusses, *for a given TPI* (i) how each of the above terms varies with core width, and (ii) the relative values of each term for CMP and SMT *for the same core width*.

(a) IPC *(SMT $\leq$ CMP for our workloads):* For both SMT and CMP, we expect IPC to increase with core complexity at a diminishing rate, eventually leveling off, as seen in Figure 4(a). For our compute-bound workloads, at a given core complexity, CMP achieves almost perfect speedup in IPC, equal to the number of threads or processor cores. The IPC speedup achieved by SMT at that core complexity depends on the superscalar IPC of the individual applications and on the resource sharing interactions among the constituent applications. The higher the superscalar IPCs and the more negative the interaction, the more difficult it is for SMT to get a high IPC speedup. Thus, for our workloads, we expect that the SMT IPC will be $\leq$ the CMP IPC at a given complexity, with the relative difference depending on the above two factors.

(b) Capacitance C *(SMT $<$ CMP)*: C increases rapidly with complexity, and depends only on the power model used. It is proportional to the *maximum power* of a processor. We see that at any core complexity, C of SMT is only a little higher than that of the superscalar, since an SMT processor adds very little hardware to the base superscalar. C for CMP, however, is a factor of N higher than that of the corresponding superscalar (and hence SMT), where N is the number of processor cores.

(c) Activity factor α *(SMT $>$ CMP)*: Since α is the fraction of total transistors switched per cycle, informally, it depends on (i) the amount of clock gating or other power management techniques in the system, and (ii) on the fraction of total transistors that are "useful" to switch (this is roughly correlated to IPC/C, since more useful switching implies higher IPC and more total transistors imply higher C). Without clock gating or other power management techniques, the value of α is always roughly the same. As clock gating and other power management techniques become more aggressive, α becomes more sensitive to IPC/C.

With increasing complexity, α will either stay roughly constant (e.g., with no clock or power gating) or will change roughly correlated to IPC/C. At lower complexities, IPC/C could increase a little, but at higher complexities, this ratio will generally go down. Figure 4(c) shows this trend.

To compare α for SMT and CMP at a given core complexity, we note that α for CMP is the same as that for the corresponding superscalar. α for SMT is higher than for the superscalar (and hence for CMP) since SMT sees a higher resource utilization. Again, the factor by which α is higher for SMT depends on the amount of clock gating (and other power management) and on the IPC speedup from SMT (relative to C) over the corresponding superscalar. For example, with no clock gating, α stays the same as the superscalar (and CMP). More aggressive clock gating and higher IPC speedup of SMT will increase the value of α relative to CMP.

(d) $\alpha\mathbf{C}$ *(SMT $<$ CMP)*: From the above discussion, we can deduce that αC increases with increasing width. Comparing CMP and SMT, αC is lower for SMT (at a given core width) due to its lower IPC and C, and high clock gating.

(e) $\mathbf{V^2 f}$ *(SMT $>$ CMP for our workloads)*: Since f is inversely proportional to IPC for a given TPI and since V also depends on f, the $V^2 f$ curve decreases rapidly with increasing complexity and IPC (following IPC^3 where $V \propto f$). At a given complexity, SMT will have a higher value of $V^2 f$ than CMP since the IPC of SMT is lower for our workloads.

(f) EPI *(SMT ? CMP)*: The EPI graph in part (f) is the product of the previous two curves (αC, $V^2 f$). For both CMP and SMT, with increasing complexity, αC generally increases. At lower complexities, this increase is offset by the decrease in $V^2 f$ (due to increasing IPC), reducing EPI. As the IPC increase diminishes, however, the decrease in $V^2 f$ is insufficient, causing EPI to increase again. Thus, each EPI vs. core complexity curve has a minimum point, which is the highest energy efficiency point mentioned above.

Putting it together – Why is CMP EPI better than SMT EPI?

For SMTN to have a better EPI than CMPN (N is the number of threads), the most energy efficient SMT configuration (call this $C_{min}SMT$) must have a higher efficiency than the most efficient CMP configuration (call this $C_{min}CMP$). Let us start by considering the relative efficiency of SMT at the $C_{min}CMP$ configuration. Based on the above discussion, at the core complexity

[4] This only considers dynamic power. Static or leakage power was negligible for the simulated technology. Although static power is expected to become more important and CMP's higher area for equal performance may seem to imply higher static power, there are several factors that make it unclear whether static power will favor CMP or SMT. First, analogous to clock gating, static power can also be contained using power gating and various process technologies (e.g., SOI, multiple-threshold transistors, etc. [4]). Second, most of the chip area is typically consumed by the L2 cache which is common to both CMP and SMT. Finally, several factors favor CMP *when comparing at equal performance* – SMT's higher average dynamic power implies higher temperature which increases leakage, SMT's slightly more complex cores require higher area, CMP's lower resource utilization implies higher potential for power gating, and CMP's lower required frequency allows for slower but lower leakage transistors.

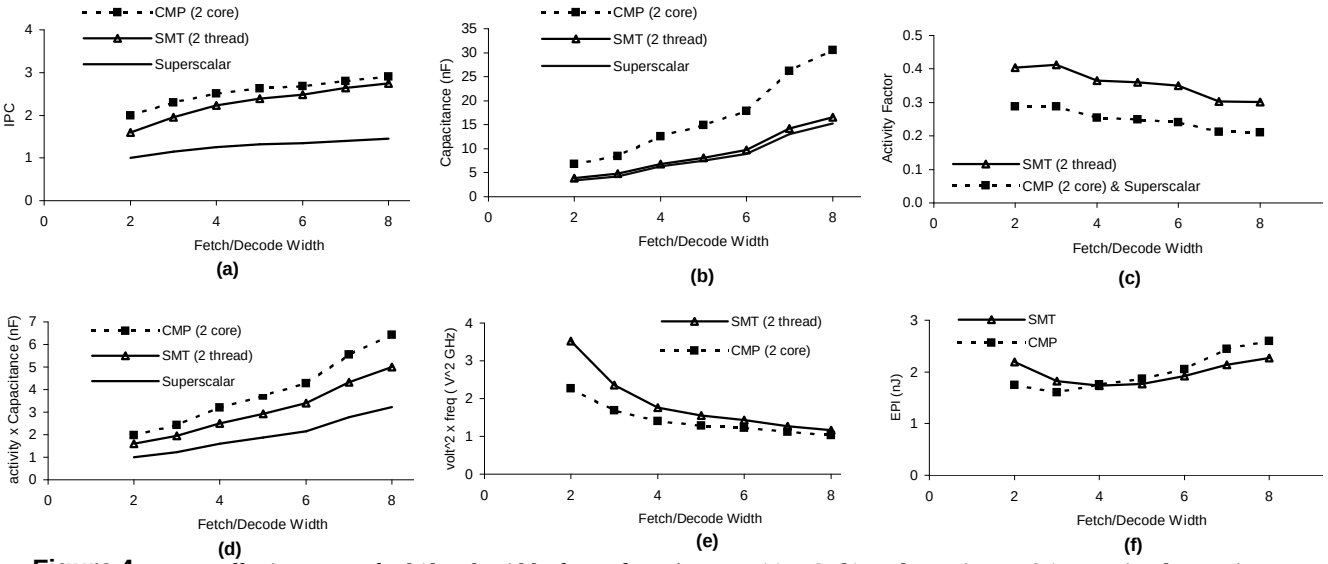


Figure 4. **Factors affecting EPI vs. fetch/decode width of core, *for a given TPI*. (a) IPC, (b) total capacitance, C (proportional to maximum power), (c) activity factor, α, (d) αC (proportional to average power), (e) $V^2 f$, and (f) EPI.**

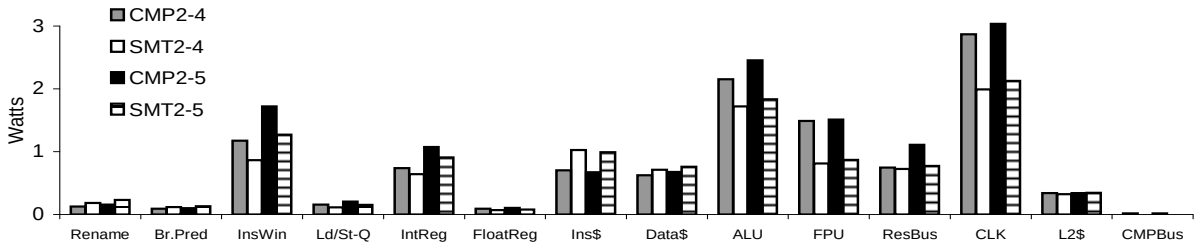


Figure 5. **Average power of individual structures for CMP2-4, SMT2-4, CMP2-5, SMT2-5 for MPGe_MPGe at the maximum frequency.**

of $C_{min}CMP$, SMT has an advantage over CMP from C, which is much lower than that for CMP. However, SMT has a disadvantage from α (based on clock gating and IPC speedup from SMT) and $V^2 f$ (based on IPC speedup from SMT and CMP). If SMT has a high enough IPC speedup (relative to CMP), then it is possible that this disadvantage is offset by the lower C, and SMT has a lower EPI than CMP at $C_{min}CMP$ (lower bounds for this IPC speedup are presented in the next section).

If the IPC speedup of SMT at $C_{min}CMP$ is not high enough, then it is still possible to see a lower EPI for SMT by changing its core complexity. By increasing the core complexity, SMT can increase its IPC and reduce $V^2 f$.[5] However, this increases C, offsetting some of the benefit of reduced $V^2 f$. C will increase by a larger amount as the slope of the C vs. complexity curve gets steeper. Increasing complexity could also increase α, depending on the amount of clock gating and the relative change in IPC to C. Depending on how much the IPC of SMT rises with respect to a rise in αC, the increase in IPC (i.e., reduced $V^2 f$) may or may not be able to beat the EPI of CMP. Since the minimum EPI point for CMP is also the minimum for the superscalar, the superscalar IPC does not rise fast enough with increasing core complexity at this point. The rise in IPC for SMT must therefore come from a high speedup of SMT over the superscalar from running multiple threads together, not just from the inherent ILP of each thread. Thus, whether SMT or CMP will have lower EPI depends on a number of factors, as summarized below.

In summary, the following factors will hinder SMT from achieving a higher energy efficiency than CMP:

1. High level of clock gating (or other power management).
2. Steep capacitance vs. core complexity curve, requiring more IPC speedup from SMT to be more efficient.
3. Workload characteristics that make it harder for SMT to obtain IPC speedup over the corresponding superscalar; e.g., high superscalar IPC or negative resource sharing interactions among constituent applications.
4. Workload characteristics that give CMP a high IPC speedup over the corresponding superscalar (e.g., compute-bound workloads).

For the systems we study, all of the above factors are present, hindering the energy efficiency of SMT relative to CMP. The first factor is not likely to change towards favoring SMT in the near future. Nevertheless, we ran experiments increasing the non-clock-gated circuitry to 20% from 10%. This made SMT slightly better than reported here, but CMP is still better for most workloads. Although unrealistic, we also experimented with eliminating clock gating altogether. This made SMT significantly better for most 2-thread workloads and for the lower performance region of 4-thread workloads but CMP was still far better for most of the 4-thread workloads in the high and medium performance regions. The second factor above is also likely to hold for out-of-order cores. The third and fourth are workload dependent, and could possibly lead to different results with different workloads.

4.3.2 Mathematical Model and Validation

We further formalize the above qualitative analysis with a mathematical model, and use the model to derive quantitative bounds on the IPC speedups that are required from SMT for it to be more energy efficient. Since it is impractical to simulate all

[5]The analysis also applies to reducing complexity, but this is not likely to improve efficiency since we start from a point that is also most energy efficient for the superscalar.

hardware configurations and workloads, our mathematical model plays an important role in increasing the applicability of our results to systems and workloads that we do not simulate. For simplicity, our model below assumes that $V \propto f$, but it can be modified for other relationships between V and f. The approximation gives EPI = $\alpha C f^3 \times TPI$. Substituting $f = \frac{1}{TPI \times IPC}$, we get $EPI = \frac{\alpha C}{IPC^3} \frac{1}{TPI^2}$. This expression is independent of frequency. It indicates that for a given system (i.e., superscalar, CMPN, or SMTN, where N is the number of threads) and a given TPI, there is one core architecture that provides the highest energy efficiency. This is the architecture that provides the lowest $\frac{\alpha C}{IPC^3}$ for that system, and we refer to that architecture as $Amin$.[6] We call the reciprocal of $\frac{\alpha C}{IPC^3}$ as the energy efficiency, or simply *efficiency (Eff)*, of a *core architecture*. Note that in Section 4.3.1, we defined efficiency for a *system configuration*, which depends on frequency. The above efficiency is for the core architecture and is independent of frequency.

Denote the IPC speedup given by CMPN (SMTN) over the corresponding superscalar as IPCSpeedCMPN (IPCSpeedSMTN). Note that the following assumes a given TPI for all cases. When not clear from the context, we will use subscripts or postfixes to indicate the system and core architecture that a specific quantity applies to. Thus, $\alpha_{CMP2,AminCMP2}$ refers to α of a CMP2 system using the core that is the most efficient for CMP2 for the given TPI. We know that $C_{SMTN,AminSMTN} \approx \frac{C_{CMPN,AminSMTN}}{N}$. Let $\alpha_{SMTN,AminSMTN} = G \times \alpha_{CMPN,AminSMTN}$ (note that α for CMP is the same as that for the corresponding superscalar). Then SMTN EPI is better than CMPN EPI if

$$Eff_{\text{CMPN,AminCMPN}} < Eff_{\text{SMTN,AminSMTN}}$$

i.e., if $Eff_{\text{CMPN,AminCMPN}} <$

$$\frac{IPCSpeedSMTN_{AminSMTN}{}^3 \times \frac{IPC_{CMPN,AminSMTN}{}^3}{IPCSpeedCMPN_{AminSMTN}{}^3}}{G \times \alpha_{CMPN,AminSMTN} \times C_{CMPN,AminSMTN} \times \frac{1}{N}}$$

i.e., if $IPCSpeedSMTN_{AminSMTN}{}^3 >$

$$\frac{Eff_{\text{CMPN,AminCMPN}}}{Eff_{\text{CMPN,AminSMTN}}} \times \frac{G \times IPCSpeedCMPN_{AminSMTN}{}^3}{N} \quad (1)$$

Since for our workloads, CMPN sees an IPC speedup of N, it follows that SMT is more energy efficient if

$$IPCSpeedSMTN_{AminSMTN}{}^3 > \frac{Eff_{\text{CMPN,AminCMPN}}}{Eff_{\text{CMPN,AminSMTN}}} \times G \times N^2 \quad (2)$$

Equations (1) and (2) clearly quantify the impact of all the four factors identified in the previous section as hindrances for SMT. Higher clock gating is represented by a higher G. The impact of the steepness of the C vs. core complexity curve is quantified by the ratio of the efficiency of CMP (and equivalently the superscalar) at AminCMP and at AminSMT. A steep C vs. core complexity curve will yield a higher ratio (recall that in the earlier discussion, "steepness" was considered relative to the increase in IPC, which is quantified by the efficiency). Finally, the workload characteristics are represented by the SMT and CMP IPC speedup terms.

We can also use the above equation to yield a lower bound on the SMT IPC speedup for SMT to be more efficient. We know that the efficiency ratio in the above equation is ≥ 1, since CMP is most efficient at AminCMP. Similarly, $G \geq 1$. Then for our workloads, equation (2) implies that at the maximum SMT efficiency point, the speedup in IPC of SMTN must be $> N^{2/3}$. For N=2, this is 1.59 and for N=4, this is 2.52. More generally, from equation (1), SMT must see an IPC speedup of at least 80% of the CMP speedup for two threads and 63% for four-threads.

Validation: To increase our confidence in our analysis and experiments, we attempted to fit our results within the above equation. In particular, we determined the lowest value of G from our experiments for a given performance point and plugged it into equation (2) to get a tighter bound on the SMT speedup. All our results were within this bound. Specifically, for the highest performance region, the $IPCSpeedSMT_{AminSMT}$ averaged 1.6 to 1.8 for 2-thread workloads and 2.1 to 3.0 for 4-thread workloads. These speedups are high and were sufficient for SMT to be comparable with CMP for many 2-thread workloads, but still lower than the predicted bound for 4-thread workloads.

4.4 Implications of Results

4.4.1 A Case for a Hybrid CMP/SMT Architecture

Our results show that HYB is significantly more energy efficient than pure SMT and close to CMP. Moreover, HYB uses only slightly more complex cores than CMP at the least energy configuration. Consequently, the hybrid architecture with two SMT cores generally needs less silicon area than a 4-core CMP, for equal performance. Moreover, such an architecture will also provide better energy for workloads that would not scale well in performance across a four core CMP. Consequently, HYB appears to be an attractive "best-of-all-worlds" solution for four-threads.

4.4.2 A Case for Adaptive Architectures and DVS

Our data shows that it is possible to pick one "overall best" core architecture for CMP and one for SMT to obtain close to optimal EPI for many cases. However, this core architecture is insufficient in the regions of maximum or minimum performance for many workloads, and in the middle performance regions for some workloads [15]. If these cases are important, then the best design would involve an adaptive processor that can change the active resources and fetch/retire width depending on the workload and desired performance/energy target (e.g., [2]). Previous control algorithms for such adaptations (e.g., [16]) could be applied in a straightforward way to CMP, but need further investigation for SMT. Further, CMP shows better potential for such techniques due to its lower utilization of resources.

Similarly, our data also shows that the lowest EPIs are obtained across a range of frequencies for CMP and SMT, supporting the use of DVS for these systems.

Finally, we note that CMP provides unique methods of adaptation that are not readily available to SMT architectures. Specifically, CMP can apply DVS and architectural adaptations independently to each core, depending on the type and amount of work to be done in each co-scheduled thread. We find that 10 out of the 14 two-thread workloads can obtain 9%-15% energy savings over the currently optimal CMP configuration, in the *High* and *Medim* performance regions, if independent DVS is available to each core.

5. Conclusions

This paper provides the first comprehensive comparison of the energy efficiency of CMP and SMT for multimedia applications on modern out-of-order general-purpose processors. SMT processors increase throughput by using resources more efficiently while CMP processors duplicate resources at the expense of low utilization. For the fairest comparison, it is important to compare energy

[6] Choosing a core architecture fixes a frequency for a given TPI. If this frequency is not supported by the system for the architecture with the highest efficiency, then for Amin, we must choose the core architecture with the next highest efficiency for which the corresponding frequency is supported.

at the same performance. Further, since different combinations of core architecture and frequency can provide a given performance but with different energy, it is important to explore a large design space and carefully pick the system pairs for comparison. We consider a wide range of architectures and frequencies to provide a large range of performance points. For each performance point, we compare the lowest energy SMT and CMP. We evaluate two-thread and four-thread multimedia workloads, derived from eight (sequential) multimedia benchmarks.

We find that across the performance spectrum, a CMP configuration is the most energy efficient for our systems, for all of our workloads. For two-threads, the difference between CMP and SMT is low, but for four-threads, it is significant. Our detailed analysis finds that several factors influence this outcome, including (1) aggressive clock gating, (2) high CMP speedup, (3) the relatively steep slope of the power vs. complexity curve in modern out-of-order processors, and (4) the inability of SMT to achieve the extremely high speedups required for it to be more efficient than CMP. It is unlikely that a modest change of several processor or technology parameters would bring significantly different results. Our analysis shows that it is necessary for SMT to obtain very high speedups (80% of the CMP speedup for two-thread workloads and 63% of the CMP speedup for four-thread workloads), or reduce clock gating significantly for SMT to become considerably better. Since it is impractical to simulate all hardware configurations and workloads, we develop a mathematical model that can encompass all the above factors. This model plays an important role in increasing the applicability of our results to systems and workloads that we do not simulate, including explicitly parallel applications.

Although our results clearly underscore the advantage of CMP for four-thread workloads, this advantage comes at the cost of silicon area. A hybrid architecture consisting of two cores with each core supporting a two thread SMT is much more energy efficient than SMT and has a lower area than CMP for equal performance. This architecture is also likely to perform better for workloads that do not scale across four CMP cores. Thus, such a hybrid architecture appears to be an attractive middle ground solution.

Finally, we find that at most performance points, one core architecture provides the best overall energy efficiency. There are, however, other performance points where other core architectures are optimal (for CMP and SMT). This motivates adaptive architectures that can deactivate parts of the core that lead to energy inefficiencies. Similar observations also motivate DVS. We also find that exploiting heterogeneity in CMP cores could further improve the CMP energy efficiency. SMT processors are not currently easily amenable to such adaptations.

There are several directions for future work. We would like to study the effect of various real-time scheduling algorithms on these systems and also explore how adaptive cores can bring more energy savings. We are also studying general-purpose architectures that are different from out-of-order superscalar processors to support multimedia applications with higher energy efficiency.

6. REFERENCES

[1] Intel Pentium-M Processor Datasheet. http://www.intel.com/design/mobile/datashts/252612.htm.

[2] D. H. Albonesi et al. Dynamically Tuning Processor Resources with Adaptive Processing. In *IEEE Computer, December* 2003.

[3] B. Bentley and R. Gray. Validating The Intel Pentium 4 Processor: Power Reduction Validation. http://www.intel.com/technology/itj/q12001/articles/art_3.htm. In *Intel Technology Journal*, 2001.

[4] S. Y. Borkar. Designing for power, http://www.intel.com/labs/features/mi04031.htm?iid=labs+mi04031.htm.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A Framework for Architectural-Level Power Analysis and Optimizations. In *ISCA*, 2000.

[6] J. Burns and J.-L. Gaudiot. Area and System Clock Effects on SMT/CMP Processors. In *PACT*, 2000.

[7] Y.-K. Chen et al. Media Applications on Hyper-Threading Technology, http://developer.intel.com/technology/itj/2002/. In *Intel Technology Journal, Vol.6, Issue 1*, February 2002.

[8] R. Gonzalez and M. Horowitz. Energy Dissipation In General Purpose Microprocessors. In *IEEE Journal of Solid-state Circuits*, September 1996.

[9] J. Griswell et al. Using a Performance Model to Estimate Core Clock Gating Power Savings. In *Workshop on Complexity-Effective Design*, 2002.

[10] L. Hammond, B. A. Nayfeh, and K. Olukotun. A Single-Chip Multiprocessor. In *IEEE Computer Special Issue on Billion-Transistor Processors*, September 1997.

[11] C. J. Hughes et al. Variability in the Execution of Multimedia Applications and Implications for Architecture. In *ISCA*, 2001.

[12] C. J. Hughes, V. S. Pai, P. Ranganathan, and S. V. Adve. RSIM: Simulating Shared-Memory Multiprocessors with ILP Processors. *IEEE Computer*, February 2002.

[13] R. Jain, C. J. Hughes, and S. V. Adve. Soft Real-Time Scheduling on Simultaneous Multithreaded Processors. In *RTSS*, 2002.

[14] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads. In *CASES*, 2001.

[15] R. Sasanka, S. V. Adve, E. Debes, and Y.-K. Chen. Energy Efficiency of CMP and SMT Architectures for Multimedia Workloads. In *UIUC CS Technical Report UIUCDCS-R-2003-2325*, 2003.

[16] R. Sasanka, C. J. Hughes, and S. V. Adve. Joint Local and Global Hardware Adaptations for Energy. In *ASPLOS*, 2002.

[17] J. Seng, D. Tullsen, and G. Z. N. Cai. Power-Sensitive Multithreaded Architecture. In *ASPLOS*, 1996.

[18] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-Theoretic Techniques and Thermal-RC Modeling for Accurate and Localized Dynamic Thermal Management. In *HPCA*, 2002.

[19] D. Tullsen et al. Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor. In *ISCA*, 1996.

[20] D. Tullsen et al. Converting Thread-Level Parallelism Into Instruction-Level Parallelism via Simultaneous Multithreading. In *ACM Trans. Computer Systems*, Aug. 1997.

[21] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proc. of the 35th MICRO*, November 2002.

[22] Y. Zhang et al. HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects. In *Tech Report CS-2003-05, Univ. of Virginia*, 2003.