# Understanding the Energy Efficiency of SMT and CMP with Multiclustering

Jason Cong, Ashok Jagannathan, Glenn Reinman, Yuval Tamir
Computer Science Department
University of California, Los Angeles, CA 90095 USA
{cong,ashokj,reinman,tamir}@cs.ucla.edu

## ABSTRACT

In this paper we study the energy efficiency of SMT and CMP with multiclustering. Through a detailed design space exploration, we show that clustering closes the energy efficiency gap between SMT and CMP at *equal performance points*. Specifically, we show that the energy efficiency of CMP compared to SMT at a given performance decreases from a maximum of 25% in a monolithic processor case to 6% when the processor resources are clustered. By carefully considering floorplans, we show that this is, in part, enabled by the small energy consumption (less than 3%) of the interconnection buses required for clustering, even with SMT. As the gap narrows, we show that the efficiency of SMT versus CMP depends on the contribution of leakage energy: at lower leakage, the CMP tends to be better than the SMT, while the SMT outperforms the CMP at higher leakage levels. We demonstrate these results over a wide range of performance and machine configurations.

## Categories and Subject Descriptors

C.1 [**Computer Systems Organization**]: Processor Architectures.

## General Terms

Performance, Design, Experimentation.

## Keywords

Energy efficiency, simultaneous multithreading, chip multiprocessing.

## 1. INTRODUCTION

Simultaneous multithreading (SMT) [1] and chip multiprocessing (CMP) [2] are two architectural approaches to exploit thread-level parallelism using available on-chip resources. SMT allows instructions from multiple threads to share several critical processor resources, thus increasing their utilization. The advantage of SMT is area-efficient throughput [3]. CMPs, on the other hand, improve system throughput by replicating processor "cores" on a single die. As both these paradigms are targeted toward multithreaded workloads, comparing their efficiency in terms of performance, power, and thermal metrics has drawn the attention of several researchers [4, 5, 6].

Unlike prior work, we explore the energy efficiency of SMT and CMP in the context of clustered processors. We do this for several reasons. First, many modern processors [7, 8] already employ some form of limited clustering. Additionally, Zyuban [9] demonstrated the energy efficiency of clustered processors when on-chip resources are scaled for increased throughput. As SMT could potentially require increasing on-chip resources to extract more parallelism, clustered processors are natural candidates to enable this with limited impact on energy. Through a detailed design space exploration, we make the following contributions in this work:

- We show that multiclustering reduces the gap in energy efficiency between SMT and CMP for a given performance. For the design space explored, we find that a monolithic CMP consumes up to 25% less energy than a monolithic SMT delivering similar performance. However, this gap is reduced to 6% once the processors are clustered. By carefully considering floorplans, we show that this is, in part, enabled by the small energy consumption (less than 3%) of the interconnection buses required for inter-cluster communication.

- As the CMP tends to use more resources than the SMT to deliver a certain level of performance, we show that the energy efficiency of SMT versus CMP depends on the leakage energy contribution. At $70nm$, the CMP is more energy-efficient than the SMT when the leakage energy is low. As the leakage energy increases (which is expected to be the case in future technologies), SMT becomes more energy-efficient than the CMP. We show these results using a conservative SMT model where the front-end resources (branch predictor, instruction cache etc.) are separate for each thread, similar to a CMP. We believe that these results should hold for more aggressive SMT models which also share these front-end resources among threads.

Based on these results, we conclude that if leakage energy cannot be kept under control in future processors, SMT allows to extract more throughput for a given amount of resources and amortize the leakage overhead across more num-


*ISLPED 2005*, August 8–10, 2005, San Diego, California.

ber of instructions. This makes SMT more energy-efficient than the CMP at higher levels of leakage energy.

The remainder of this paper is organized as follows: Section 2 discusses prior work related to this research and highlights the main differences. We present the baseline processor model in Section 3 and also discuss performance and energy metrics. Our delay and energy modeling for blocks and wires is presented in Section 4. Section 5 presents our simulation methodology, and experimental results are discussed in Section 6. We conclude the paper in Section 8 with some future directions.

## 2. PRIOR WORK

Several prior studies [5, 4, 3] have examined the energy efficiency of SMT and CMP under different workloads. Kaxiras et al. [5] studies the problem for mobile workloads using VLIW cores and show the SMT to be more efficient. Sasanka et al. [4], on the other hand, explores the problem for multimedia workloads, and conclude that CMP is better than SMT. However, it is not clear how these results apply to other classes of benchmarks. Recent work by Li et al. [3] explores the energy efficiency of SMT for general-purpose workloads and shows that it offers superior performance in terms of the energy-delay-square metric. Li et al. [6] also study the energy and thermal properties of SMT and CMP in the context of a fixed die size and show the CMP to be superior for CPU-bound benchmarks, and SMT to be better on memory-bound benchmarks due to larger L2 cache.

None of the aforementioned studies has considered SMT and CMP evaluation with clustering, which we consider in this work. Our methodology resembles the work by Sasanka et al. [4] in the sense that we also compare the SMT and CMP at *equal performance points*, ignoring the impact on area (which we think will manifest itself in the form of energy).

## 3. PROCESSOR MODEL AND METRICS

Our baseline SMT model with two clusters and two thread contexts is shown in Figure 1, and the corresponding processor parameters are listed in Table 1. While prior research [1] has looked at sharing front-end resources among threads, we assume that each thread has its own branch predictor and instruction cache. Thus, only the back-end execution resources such as issue queue entries, physical register file, functional units, and data cache are all shared by the threads similar to [10]. We use the Advanced RMBS steering heuristic [11] to steer instructions to different clusters. However, determining the number of ready instructions [11] in each issue window is not a straightforward task in the presence of wire latencies. Therefore, we use the number of instructions dispatched to each cluster as a measure of load imbalance. We use the ICOUNT [12] thread selection policy to prioritize fetching instructions from different threads.

In our clustered model, we assume that the physical register file is split among the clusters. Thus, copy instructions are required to transfer register data between clusters. We use a single-issue, six-entry copy instruction queue at each cluster for this purpose. Similarly, each cluster is provided with a small set of buffers to receive register data from other clusters. Register values are communicated on demand so that energy consumption due to data transfer between clusters can be minimized.

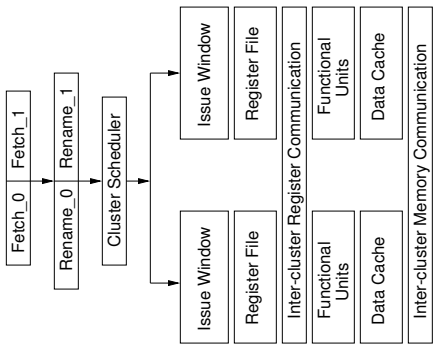



**Figure 1: Baseline clustered SMT processor with two thread contexts and two clusters.**

Another important aspect of clustered processors is the first-level data cache design. While different strategies [9, 13, 14] have been studied from a performance standpoint, we have limited our study to a model where the cache banks are replicated at each cluster, primarily because of the following reasons: (1) while providing better performance than other distributed cache models, this enables load instructions to be steered to any cluster and eliminates the need for address prediction and misprediction recovery issues [9]; and (2) it makes energy modeling simpler.

| | |
|---|---|
| Fetch | 4K entry 4-way associative BBTB<br>32KB g-share |
| I-cache | 32KB, 2-way |
| Decode Width | 4 |
| Issue Width | 4 |
| Issue Window | 16 |
| ROB | 256 entries per thread |
| LSQ | 128 entries |
| Register File | 80 INT and 80 FP |
| Functional Units | 4 INT, 2 FP |
| D-cache | Replicated 32KB, 4-way, 32 bytes/block<br>1RW port per cluster<br>1 cycle latency for access |
| Unified L2 | 2MB, 8-way, 64 bytes/block<br>12 cycle latency<br>Shared bus to DL1 |
| Main memory | 180 cycles for first chunk |

**Table 1: Baseline processor parameters used in this study.**

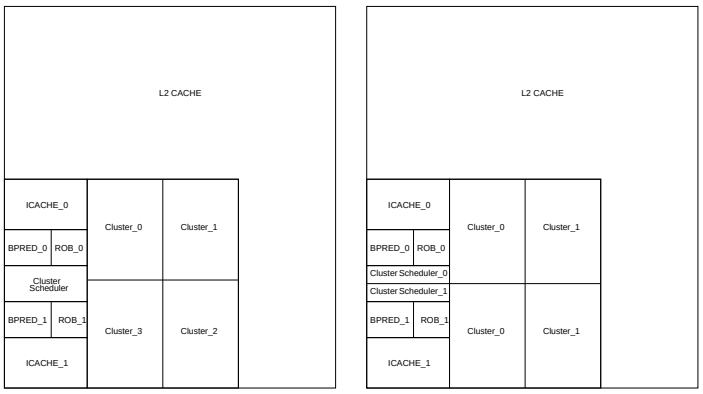
For the purpose of correct memory disambiguation, store instructions are allocated entries in the LSQs of all the clusters, similar to Zyuban's [9] proposal. However, only one copy of the store instruction is allowed to execute. Store addresses and data are broadcast to all clusters so that memory disambiguation can be done at each cluster locally. This also allows to maintain cache coherency.

Our inter-cluster communication model consists of two separate buses: a 72-bit wide (64-bit data and 8-bit control) *register data communication bus* and a 32-bit wide *memory address communication bus*. We use a ring topology to interconnect the clusters, and the latency of each link on the ring depends on the floorplan. To avoid bus arbitration issues, we assume that there are as many buses as the number of clusters so that each cluster is able to freely broadcast data at any time.

### Metrics

Several metrics [15] have been proposed to evaluate the performance of multi-threaded workloads. We measure the performance of SMT and CMP using the weighted IPC metric proposed by Snavely et al. [15]. The speedup of a $N$-thread workload is calculated as

$$\text{SMT speedup} = \frac{1}{N} \sum \frac{IPC_{SMT}(i)}{IPC_{non-SMT}(i)}$$




(a) SMT (b) CMP

**Figure 2: Floorplans for a four-cluster SMT (a), and a corresponding two-core CMP (b). Each core on the CMP has two clusters, and a separate cluster scheduling logic.**

where $IPC_{SMT}(i)$ is the IPC of the $i$'th thread during the SMT execution and $IPC_{non-SMT}(i)$ is its IPC during a single thread execution. We choose this metric because it considers how each individual thread performs under SMT relative to its non-SMT performance. All our speedups are calculated relative to the IPC of each benchmark on the baseline, non-clustered, non-SMT machine.

In contrast, we use the total energy consumed during each run for the energy metric. We use the energy-delay product as a metric for comparing different architectures. This can be expressed as follows:

$$E \times D = \frac{Energy}{Instruction} \times \frac{Cycles}{Instruction} = \frac{Energy/Cycle}{IPC^2}$$

One possible way to visualize the solutions based on the $E \times D$ metric is to plot them on an energy-per-cycle vs. IPC graph. Due to the quadratic dependency on IPC, points with equal energy-delay product appear as parabolas on this graph. Thus, solution points on this graph which lie on lower equal energy-delay product curves represent configurations with a lower energy-delay product – i.e., better design points. For a better understanding of these curves, we point the reader to [9].

# 4. DELAY AND ENERGY MODELING

We model the delay and energy of the major components of the processor based on CACTI [16] and Wattch [17] respectively. However, unlike Wattch, we model a separate ROB, issue window, and a physical register file. We assume aggressive clock gating [17] for all the architecture configurations that we study. As it is not possible in practice to achieve 100% clock gating, we assume that only 90% of the circuit power can be turned off with clock gating, as in typical industrial circuits [17]. We model leakage energy as a fraction of the total *unconstrained* dynamic energy, similar to the work by Li et al. [3].

The main interconnection buses that may contribute to the energy in a clustered processor over a monolithic processor are (a) the *instruction dispatch* bus, (b) the *register data* communication bus, and (c) the *memory address* communication bus. In order to estimate the contribution of energy by these buses, we need to estimate the lengths of the wires which can only be obtained from a floorplan. Since searching all possible layouts is an expensive operation, we assume a similar floorplan for all configurations with a given number of clusters and estimate the interconnect lengths in each case based on this floorplan. A sample layout for a four-cluster SMT and the corresponding CMP is shown in Figure 2.

For the purpose of floorplanning, many of the block areas were estimated using CACTI [16], while the functional unit areas were estimated based on the data used by Palacharla et al. [18]. For the architecture configurations that we explored in this study, the total interconnection bus length across a specific cluster was between 210$\mu$m and 1800$\mu$m. We calculate the latency of these buses based on interconnect performance estimation models by Cong and Pan [19], which considers different interconnect optimization techniques such as wire sizing and buffering. The latency is obtained as the minimum number of cycles required to cover the wire delay, and the energy is calculated based on the wire and buffer parasitics.

# 5. SIMULATION METHODOLOGY

The performance simulator used in this study is based on the SimpleScalar 3.0 toolset [20]. The simulator models an out-of-order clustered processor such as Alpha 21264 [8] and runs Alpha AXP ISA executing only user-level instructions. The simulation is event driven, including execution down any speculative path until the detection of a fault, TLB miss, or branch misprediction. We assume a 70$nm$ process technology for all experiment in this study, and fix the target clock frequency at 4GHz, which is representative of next-generation superscalar processors. Since we explore changes in several architectural parameters, we assume the blocks and wires can be pipelined to any depth, as required at the target frequency.

For this study we use 12 SPEC2000 benchmarks, six integer (`gzip, vpr, twolf, bzip2, perl` and `eon`) and six floating point (`wupwise, art, sixtrack, mesa, ammp` and `lucas`) programs. We chose these benchmarks based on their IPC (high and low), their response to scaling back-end resources in the processor (compute-intensive or not) and their memory requirements (large or small footprints). We then create 12 pairs of two-thread workloads such that each benchmark appears the same number of times in the entire workload. The IPC numbers presented here are averages over all the benchmarks. We use the SimPoint toolset [21] to get representative simulation points, and simulate each two-thread workload for 200 million instructions for every architecture configuration.

Before we compare the energy efficiency of SMT and CMP at a certain level of performance, it is necessary to find the energy-efficient processor configurations for the SMT and CMP separately. Toward this end, we follow a methodology similar to that of Zyuban [9]. Basically, we perform several simulations for each multi-thread workload by varying important processor parameters, and plot the results on a energy-per-cycle vs. IPC graph. The lower convex hull (LCH) of these points provides a set of non-dominated solutions – i.e., the lowest energy solution for a particular level of performance. We can then directly compare these LCH curves for different architectures to understand their energy efficiency. A similar approach was also followed by Sasanka et al. [4].

While there are several parameters that affect the performance and energy of a given architecture, we primarily concentrate on the efficiency of back-end resource sharing enabled by SMT. Hence, we choose to generate a variety of architectural configurations by varying only the following: issue window, number of entries in the physical register file, number of clusters, and the number of functional units and memory ports to the cache. The range of values explored for these parameters is shown in Table 2.

| | |
|---|---|
| Issue Width | 4, 6 and 8 |
| Number of clusters | 1, 2, and 4 |
| Issue Window | 8 to 128 |
| Physical RF | 72 to 256 |
| L1 memory ports | 1 and 2 |

**Table 2: Parameters that were varied to generate the energy-efficient configurations for a given architecture.**

## 6. RESULTS

In Figure 3 and Figure 4 we show the family of energy-efficient architecture configurations for four different architectures (monolithic SMT, monolithic CMP, clustered SMT, and clustered CMP) and three different issue widths (four, six, and eight). Note that a four wide CMP implies two cores each with an issue width of two. These curves were obtained by simulating approximately 700 configurations for the monolithic case, and 350 for each clustered architecture. For these experiments, we assume a fixed L2 cache size of 2MB, and only change the core parameters as mentioned in Table 2. We also set the leakage energy to be 10% of the unconstrained dynamic energy, as in [3].

Comparing the monolithic SMT and CMP architectures in Figure 3, the CMP appears more energy efficient than the SMT over a wide range of performances. The efficiency of CMP increases with the issue width, and the CMP can save up to 25% energy compared to the SMT at an issue width of eight. While Li et al. [6] shows the SMT to be superior to CMP for memory-bound benchmarks, the study used a smaller L2 cache for the CMP compared to the SMT. As we use the same L2 cache size for all our configurations, we find the CMP to be more efficient for the range of IPCs we explored.

However, for a fixed amount of resources, the SMT delivers a higher performance than the CMP. This can be seen by comparing the leftmost and rightmost points on the corresponding SMT and CMP curves, which have almost identical resources. At all issue widths, the maximum throughput achieved by the SMT machine is higher than that obtained by an equivalent width CMP. This underlines the effective dynamic resource sharing enabled by SMT. The increased throughput for the SMT comes with an increase in the energy consumption due to the higher utilization. For instance, the maximum performance for the eight-issue SMT is around 9.5% more than the maximum performance of the eight-issue CMP with identical resources, but this improvement almost doubles the consumed energy.

Comparing Figure 3 and Figure 4, we can see that clustering can reduce the energy consumption by up to 50% at a given performance. We can also see that the energy savings increase as the issue width of the machine increases: the maximum energy reduction is around 25% for the 4-wide machine, 40% for the 6-wide, and almost 50% for the 8-wide. A similar result for single thread workloads was shown by Zyuban [9].

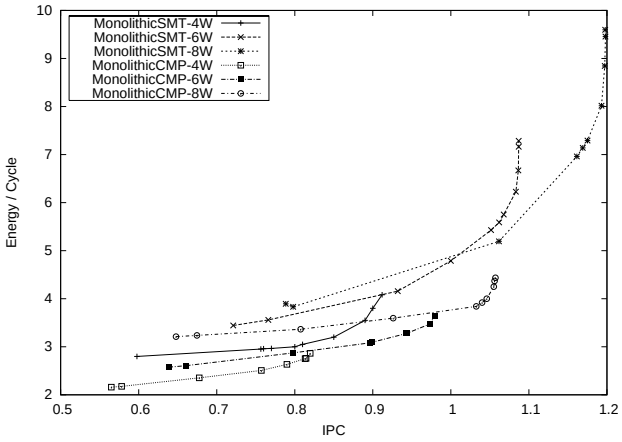



**Figure 3: Energy efficient families for monolithic SMT and CMP processors.**

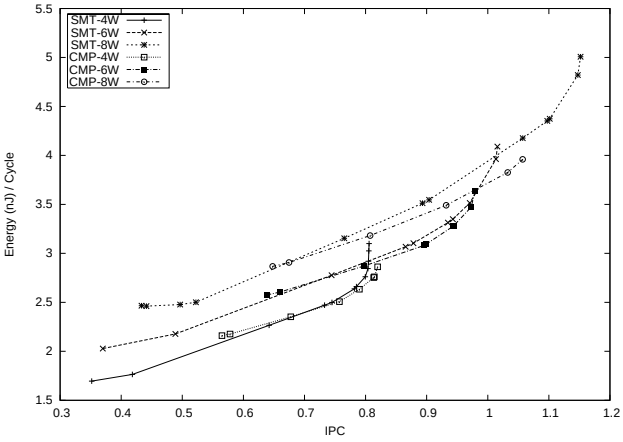



**Figure 4: Energy-efficient families for clustered SMT and CMP processors.**

One reason for this energy benefit is that the interconnect energy overhead due to clustering is very minimal in all the architectural configurations we explored, even with SMT. The wires, together with the copy instruction queue and the data buffers, contributed to less than 3% of the overall energy of these configurations. Notice that this is even true when the number of clusters is four, which has more interconnect energy associated with the buses. After

careful examination, we found that the wires that span a cluster in our experiments were only between 210$\mu$m and 1800$\mu$m. However, at 70$nm$ technology, a wire has to be nearly 2600$\mu$m to see a benefit from buffer insertion [19]. As the wirelengths in our study could be accommodated within the target frequency without any buffering, this significantly reduced the energy contribution from the interconnects.

Interestingly, the energy efficiency of SMT and CMP is only marginally different with multiclustering, as shown in Figure 4. While Figure 4 and Figure 3 show that clustering helps to reduce energy for both the SMT and CMP, the static partitioning of CMP means that it is already clustered at a coarser granularity, and this limits the gain for the CMP from multi-clustering. Thus, at issue widths of four and six, we find the SMT and CMP to be similar in terms of their efficiency, while at a width eight, the CMP is more efficient by up to 6%. Correspondingly, we see the gap between SMT and CMP efficiency reduced from a maximum of 25% to 6% once the processor resources are clustered. As the trend is similar over a wide range of IPCs, we believe that this result will hold even for wider machine widths and more threads.

# 7. IMPACT OF LEAKAGE ON ENERGY EFFICIENCY

As mentioned earlier, we observed that the CMP typically requires more resources (hence area) than the SMT to achieve the same level of performance. For instance, in Figure 4, an IPC of just over 1.0 can be achieved in the case of CMP with a eight-wide machine, whereas a machine width of six is sufficient for the SMT case. However, we find that the CMP consumes less energy than the SMT in Figure 4. This is possible because the dynamic energy is the dominant component of the total energy under our leakage assumptions, and hence the increased resource utilization of SMT makes it less energy efficient. Intuitively, the increased area of the CMP will affect its energy efficiency only when the leakage energy due to these extra transistors increases. As leakage energy is expected to grow to more than 50% of the total energy [22], we show the sensitivity of the energy efficiencies of SMT and CMP to the leakage energy. These results are shown in Figure 5, where we increase the leakage from 10% to 40% of the unconstrained dynamic energy (for reference, leakage power contributes to almost 40% of the power consumption of a Pentium 4 [22] processor).

In general, it can be observed from Figure 5 that the curves for the CMP tend to move up slightly faster than the corresponding SMT curves as the leakage energy contribution increases. This makes the CMP become less energy efficient at performance points where it was more efficient than the SMT at lower leakage. Even in the case of a 10% leakage factor, we see that the SMT consumes marginally lower energy than the CMP for a four-wide machine around an IPC of 0.57. Though both machines have a width of four, each core on the CMP uses significantly more entries in the issue window and physical register file in each core to deliver the same performance at a narrow width of two. The leakage contribution due to these larger structures make the CMP less efficient, even for a 10% leakage contribution. As the leakage increases to 30%, we find that the SMT consumes approximately 10% lesser energy than the CMP for this performance point.

Thus, when the contribution of leakage energy increases, we find and increase in the range of IPC values within which the SMT becomes more efficient than the CMP. This trend can be observed at every issue width in the curves in Figure 5. While the CMP is better by, at most, 6% in terms of energy for a leakage of 10%, the SMT becomes more efficient by up to 10% at a 40% leakage contribution. Notice that these results are obtained on a conservative SMT model where we assume that each thread has a dedicated front-end much like a CMP. For more aggressive SMT models where the front-end is shared among multiple threads, we think that the SMT can save more than 10% energy at higher leakage levels compared to a CMP for the same level of performance.

These results show that if leakage energy cannot be controlled effectively and it contributes significantly to the total energy consumption, it is useful to use SMT to exploit thread-level parallelism to increase the throughput and amortize the leakage cost over a larger number of instructions.

# 8. CONCLUSION

We demonstrate several interesting results through this work. First, we show that multiclustering helps to reduce the gap in the energy efficiency of SMT and CMP at equal performance points. For the sample design space explored, we show that while the CMP is up to 25% more energy efficient than the SMT for monolithic cores, clustering reduces the CMP efficiency to just 6%. We find that this is, in part, enabled by the limited energy consumption (less than 3%) of the interconnection buses required for cluster communication. Finally, we show that the energy efficiency of SMT and CMP depends on leakage – CMP is more efficient with dominant dynamic energy, while SMT tends to become efficient with increasing leakage energy. Thus, we conclude that when leakage energy is significant, it is useful to extract more throughput using SMT so that the leakage cost can be amortized over a larger number of instructions.

# 9. REFERENCES


[1] D. Tullsen, S. Eggers, and H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," in *Proceedings of the 22rd Annual International Symposium on Computer Architecture (ISCA)*, June 1995.

[2] K. Olukotun, B. A. Nayfeh, L. Hammond, K. Wilson, and K. Chang, "The Case for a Single-Chip Multiprocessor," *SIGOPS Oper. Syst. Rev.*, vol. 30, no. 5, pp. 2–11, 1996.

[3] Y. Li, D. Brooks, Z. Hu, K. Skadron, and P. Bose, "Understanding the Energy Efficiency of Simultaneous Multithreading," in *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, pp. 44–49, 2004.

[4] R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes, "The Energy Efficiency of CMP vs. SMT for Multimedia Workloads," in *Proceedings of the 18th Annual International Conference on Supercomputing*, pp. 196–206, 2004.

[5] 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 *Proceedings of the 2001 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems*.

[6] Y. Li, K. Skadron, Z. Hu, and D. Brooks, "Performance, Energy, and Thermal Considerations for SMT and CMP Architectures," in *Proceedings of the Eleventh IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2005.

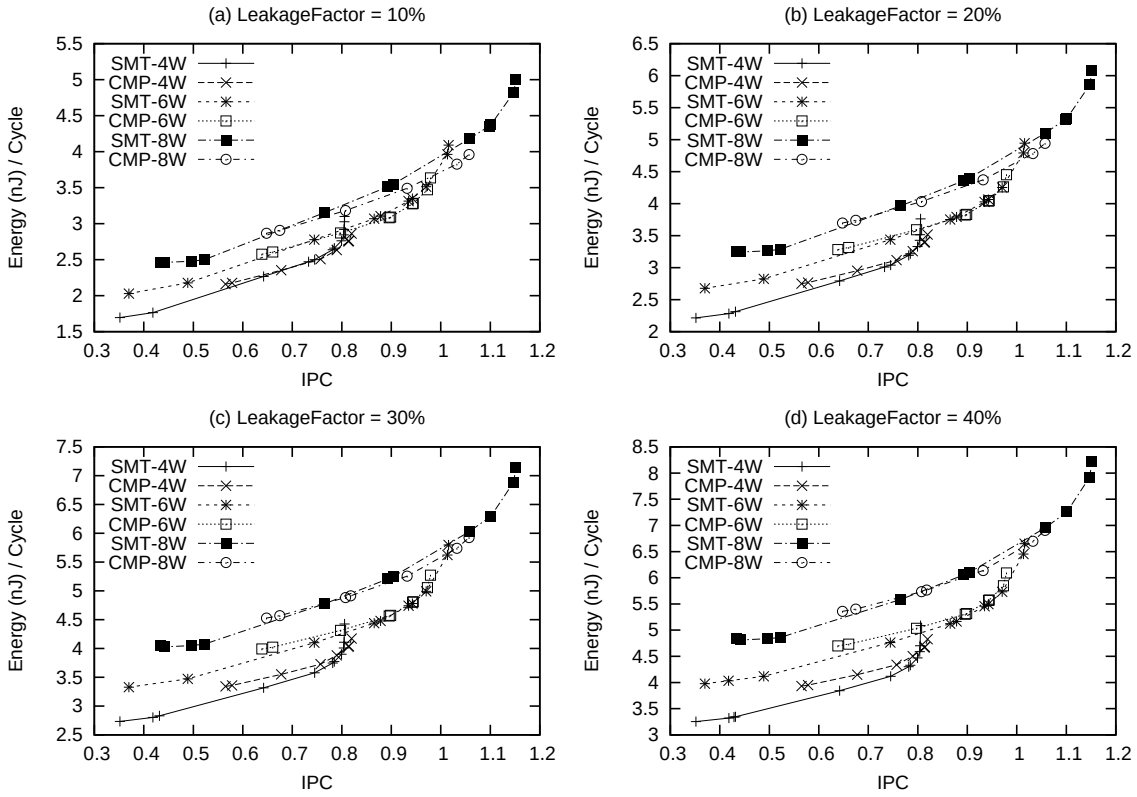



**Figure 5: Comparison of energy-efficient configurations for SMT and CMP for different levels of leakage energy. CMP tends to become less energy efficient as the leakage energy increases due to extra area associated with the resources to produce similar performance.**


[7] G. Hinton, D. Sager, M. Upton, D. Boggs, D. Carmean, A. Kyker, and P. Roussel, "The Microarchitecture of the Pentium 4 Processor," *Intel Technology Journal Q1*, 2001.

[8] R. Kessler, E. McLellan, and D. Webb, "The Alpha 21264 Microprosessor Architecture," in *International Conference on Computer Design*, Dec. 1998.

[9] V. V. Zyuban and P. M. Kogge, "Inherently Lower-Power High-Performance Superscalar Architectures," *IEEE Transactions on Computers*, vol. 50, no. 3, pp. 268–285, 2001.

[10] F. Latorre, J. Gonzalez, and A. Gonzalez, "Back-end Assignment Schemes For Clustered Multithreaded Processors," in *Proceedings of the 18th Annual International Conference on Supercomputing*, pp. 316–325, 2004.

[11] R. Canal, J.-M. Parcerisa, and A. Gonzalez, "Dynamic Code Partitioning for Clustered Architectures," *Int. J. Parallel Program.*, vol. 29, no. 1, pp. 59–79, 2001.

[12] D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor," in *ISCA '96: Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pp. 191–202, 1996.

[13] P. Racunas and Y. N. Patt, "Partitioned First-Level Cache Design for Clustered Microarchitectures," in *Proceedings of the 17th Annual International Conference on Supercomputing*, pp. 22–31, 2003.

[14] R. Balasubramonian, S. Dwarkadas, and D. H. Albonesi, "Dynamically Managing the Communication-Parallelism Trade-Off in Future Clustered Processors," in *Proceedings of the 30th Annual International Symposium on Computer architecture*, pp. 275–287, 2003.

[15] A. Snavely and D. M. Tullsen, "Symbiotic Jobscheduling for a Simultaneous Multithreading Processor," in *In Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, Nov. 2000.

[16] S. Wilton and N. Jouppi, "An Enhanced Access and Cycle Time Model for On-Chip Caches." Compaq WRL TR-93-5, July 1994.

[17] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural-Level Power Analysis and Optimizations," in *27th Annual International Symposium on Computer Architecture*, June 2000.

[18] S. Palacharla, N. P. Jouppi, and J. E. Smith, "Complexity-Effective Superscalar Processors," in *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pp. 206–218, June 1997.

[19] J. Cong and D. Z. Pan, "Interconnect Estimation and Planning for Deep Submicron Designs," in *DAC '99: Proceedings of the 36th ACM/IEEE Conference on Design Automation*, pp. 507–510, 1999.

[20] D. C. Burger and T. M. Austin, "The SimpleScalar Tool Set, Version 2.0," Technical Report CS-TR-97-1342, U. of Wisconsin, Madison, June 1997.

[21] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically Characterizing Large Scale Program Behavior," in *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 45–57, 2002.

[22] T. Karnik, S. Borkar, and V. De, "Sub-90nm Technologies: Challenges and Opportunities for CAD," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 203–206, 2002.