Physically Constrained Architecture for Chip Multiprocessors

A Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy

Computer Science

by

Yingmin Li

August 2006

© Copyright August 2006

Yingmin Li

All rights reserved

Approvals

This dissertation is submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Computer Science

Yingmin Li

Approved:

Kevin Skadron (Advisor)

Jack W. Davidson (Chair)

Mircea R. Stan

David M. Brooks

Sudhanva Gurumurthi

Accepted by the School of Engineering and Applied Science:

James H. Aylor (Dean)

August 2006

Abstract

Recent product announcements show a clear trend towards aggressive integration of multiple cores on a single chip. This kind of architecture is called a “chip multiprocessor” or CMP. By taking advantage of thread level parallelism, CMP can achieve better performance/power scalability with technology than single core architectures. However, this trend presents an expansive design space for chip architects, encompassing number of cores per die, core size and complexity (pipeline depth and superscalar width), core type (in-order and out-of-order, single-threaded or multi-threaded), memory hierarchy and interconnection fabric design, operating voltage and frequency, and so on. These choices are especially difficult because all the variables of interest are inter-related and must be considered simultaneously. Furthermore, trade-offs among these design choices vary depending both on workloads and physical constraints like power, area and thermal constraints. Ignoring any of these physical constraints at early design stage may lead to significant performance loss.

In this dissertation I explore this multi-dimensional design space across a range of possible physical constraints, for multiple categories of workloads. To assist this design space exploration, a validated systematic infrastructure is designed to help accelerate CMP simulation. I believe this is the first work which considers so many cores, optimizes across so many design variables simultaneously and is aware of so many important physical constraints.

Acknowledgments

I want to thank my advisor Kevin Skadron. His guidance, encouragement, and support is a vital part of this work. I am truly grateful to his sincere, straightforward, insightful and patient advising. I also want to thank Prof. David Brooks, who has effectively been my co-advisor for this work. Many thanks to the dissertation committee, Prof. Jack Davidson, Prof. Mircea Stan and Prof. Sudhanva Gurumurthi. They have spent enormous time reviewing this work.

My dissertation work benefits a lot from the collaboration with Prof. David Brooks' research group at Harvard and with Zhigang Hu at IBM research.

I should thank my family, esp. my mother Peilan Zhang. Her vision brought me here from a small village in China. It is hard to imagine I can reach this step without her support.

Finally, I would like to thank the computer staff in our department, for their diligent work in OS installation, machine maintenance, and software installation, which makes this research work possible.

Funding support for this dissertation was provided by several grants from the NSF, the IBM research and Intel MRL. The National Science Foundation provided support under grant nos. CCR-0082671, NSF CAREER award CCR-0133634, CCR-0306404, and EIA-0224434. An IBM faculty award and an Intel MRL grant also supported a large portion of this dissertation.

Contents

1 Introduction **1**

1.1 Background . . . 3

1.2 Research Issues and Dissertation Overview . . . 12

2 Experimental Methodology **15**

2.1 Simulator . . . 15

2.2 Benchmark . . . 21

2.3 Speedup Metric . . . 21

3 The Energy Efficiency of Simultaneous Multithreading **23**

3.1 Introduction . . . 23

3.2 Benchmark Methodology . . . 24

3.3 Results . . . 24

3.4 Future Work and Conclusions . . . 31

4 Power and Energy Efficiency of Different Clock Gating Styles **32**

4.1 Introduction . . . 32

4.2 Modeling Methodology . . . 34

4.3 Results . . . 37

4.4 Future Work and Conclusions . . . 42

5 Performance, Energy and Temperature Considerations for CMP and SMT architectures **43**

5.1 Introduction 43
5.2 Related Work 45
5.3 Modeling Methodology 47
5.4 Baseline Results 49
5.5 Aggressive DTM constrained designs 56
5.6 Future Work and Conclusions 63

6 CMP Design Space Exploration **66**
6.1 Introduction 66
6.2 Related Work 67
6.3 Experimental Methodology 70
6.4 Results 82
6.5 Future Work and Conclusions 92

7 The Investigation of Core Type Choices for CMP **94**
7.1 Introduction 94
7.2 Related Work 96
7.3 Methodology 98
7.4 Results 103
7.5 Future Work and Conclusions 114

8 Conclusions and Future Direction **115**
8.1 Dissertation Summary 115
8.2 Future Work 117

A Glossary **121**

Bibliography **123**

List of Figures

1.1 A typical CMP organization 4

1.2 Coarse-grain multithreading vs. simultaneous multithreading 8

2.1 Modeled Processor Organization [52] 16

2.2 Floorplans used for thermal simulation. The SMT core is 12% larger than the ST core shown above. [54] 19

3.1 Performance of SMT vs. ST [52] 25

3.2 Power Dissipation of SMT vs. ST [52] 26

3.3 Energy-Delay^2 of SMT vs. ST [52] 27

3.4 Power Dissipation Breakdown by Units [52] 28

3.5 Impact of Leakage on Power Dissipation of SMT vs. ST [52] 29

3.6 Impact of PowerFactor on Energy-Delay^2 of SMT vs. ST [52] 30

4.1 Abstract diagrams of valid-bit gating [51]. 35

4.2 Abstract diagrams of stall gating [51]. 36

4.3 Abstract diagrams of array gating [51]. 36

4.4 The peak temperature of each benchmark with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51] 38

4.5 The average unit power of integer benchmarks (left) and floating point benchmarks (right) [51] 39

4.6 The average unit utilization of integer benchmarks (left) and floating point benchmarks (right) [51] 39

4.7 **The temperature of the units for integer benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]** 40

4.8 **The temperature of the units for floating point benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]** . . 41

4.9 **The temperature of FXQ for four benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]** 41

5.1 **Performance of SMT and CMP for memory-bound benchmarks (the categorization is done with 2MB L2 cache size for ST) with different L2 cache size [54]** 50

5.2 **Performance and Energy efficiency of SMT and CMP compared to ST, for low L2 cache miss workloads (left) and high L2 cache miss workloads (right) [54].** 51

5.3 **Performance and Energy efficiency of SMT and CMP compared to ST as L2 size changes. On the left are results for a benchmark (mcf+mcf) which is memory bound for all L2 configurations shown. On the right are results for a benchmark (mcf+vpr) which ceases to be memory-bound once L2 size changes from 1MB to 2MB for CMP [54].** 51

5.4 **Temperature of SMT and CMP vs. ST [54]** 53

5.5 **Temperature Difference between CMP and SMT for different technologies [54]** . . . 55

5.6 **Performance loss from DTM vs. peak temperature. Peak temperature here is plotted as the number of degrees by which the maximum temperature exceeds the trigger threshold [54]** 56

5.7 **Performance of SMT and CMP vs. ST with different DTM policies, all with threshold temperature of** $83°$**C. Workloads with low L2 cache miss rate are shown on the left. Workloads with high L2 cache miss rate are shown on the right [54].** 59

5.8 **Energy-efficiency metrics of ST with DTM, compared to ST baseline without DTM, for low-L2-miss-rate workloads (left) and high-L2-miss-rate workloads (right) [54].** . 61

5.9 **Energy-efficiency metrics of SMT with DTM, compared to ST baseline without DTM, for low-L2-miss-rate benchmarks (left) and high-L2-miss-rate benchmarks (right) [54].** 61

5.10 **Energy-efficiency metrics of CMP with DTM, compared to ST baseline without DTM, for low-L2-miss-rate benchmarks (left) and high-L2-miss-rate benchmarks (right) [54].** 62

6.1 **The validation of Zauber model [53]** 75

6.2 **Simplified temperature model validation [53]** 79

6.3 **Performance of various configurations with chip area constraint at 400mmˆ2 (without thermal control) [53]** 84

6.4 **Performance of various configurations with chip area constraint at 400mmˆ2 (R = 0.1 heat sink) [53]** 84

6.5 **Performance of various configurations with chip area constraint at 400mmˆ2 (R = 0.45 heat sink) [53]** 84

6.6 **The difference from the optimal when no thermal consideration is made at early design [53]** 89

7.1 **Normalized BIPS/Area and BIPS^3/(W.Area) with different cache size** 101

7.2 **Single thread latency (y-axis) vs. Total throughput (x-axis). Best performance towards the origin.** 104

7.3 **SpecJBB with 400sqmm die and LR heatsink** 105

7.4 **SpecJBB with 400sqmm die and HR heatsink** 106

7.5 **MCF with 400sqmm die and LR heatsink** 107

7.6 **MCF with 400sqmm die and HR heatsink** 107

7.7 **Pin-Bandwidth Constraints with 400sqmm die and LR heatsink. Each point represents one of 24, 48, or 96GB/s total chip bandwidth. The 48GB/s point is always the middle point in groups of three.** 108

7.8 **Area sensitivity (specJBB). Each point on the same line represents scaling IO core area down to 50%, 70%, and 90%.** 111

7.9 **Area sensitivity (MCF). Each point on the same line represents scaling IO core area down to 50%, 70%, and 90%.** 112

7.10 **The performance loss using optimal tradeoff configurations** 112

7.11 **Turning off one thread in SMT (points lower on the y-axis are with SMT disabled)** . . 113

List of Tables

1.1 **Configurations of commercial CMP chips (OO = Out-of-Order; IO = In-Order)** . . . 5

1.2 **ITRS roadmap** . . . 10

2.1 **Configuration of simulated processor** . . . 17

4.1 **Number of ports modeled. For clustered structures, we only report ports for a single instance [51].** . . . 34

5.1 **Categorization of integer benchmarks [54]** . . . 48

5.2 **Categorization of floating point benchmarks [54]** . . . 49

6.1 **Latencies for 19FO4 (cycles) [53]** . . . 71

6.2 **Resource Sizes with Width Scaling [53]** . . . 72

6.3 **Energy Scaling [53]** . . . 73

6.4 **Pipeline Width Area Scaling [53]** . . . 80

6.5 **Optimal Configurations for BIPS with Varying Pipeline Width, Fixed Depth (18FO4) [53]** . . . 83

6.6 **Optimal Configurations for BIPS^3/W with Varying Pipeline Width, Fixed Depth (18FO4) [53]** . . . 85

6.7 **Optimal Configurations for BIPS with Varying Pipeline Depth, Fixed Width (4D) [53]** 86

6.8 **Optimal Configurations for BIPS^3/W with Varying Pipeline Depth, Fixed Width (4D) [53]** . . . 86

7.1 **Terminology** . . . 99

7.2 **Core area of different architectures at 19FO4** (mm^2) 100

7.3 **Area of non-core structures on a CMP chip** (mm^2) 101

7.4 **Description of benchmarks** 103

7.5 **Configurations to achieve the best average performance across benchmark** 113

Chapter 1

Introduction

This dissertation explores the broad design space of Chip Multi-Processor (CMP) or "multi-core" architectures subject to power consumption, chip temperature, pin-bandwidth, single-thread performance and chip area constraints and develops modeling tools suitable for carrying out this exploration. The central thesis of this work is that failure to consider these constraints jointly at an early design stage will lead to vastly inferior designs. The design space for CMP is highly multidimensional. All these design dimensions and physical constraints are inter-dependent, and this dissertation features a joint optimization of all design dimensions subject to physical constraints.

Recently two clear trends motivating CMP architectures have developed in industry. The first is an increased focus on thread level parallelism (TLP), with a gradual de-emphasizing of instruction level parallelism (ILP) in chip designs. Second, the exponential growth of chip frequency is finally slowing down. This is due to a number of fundamental reasons:

Given a constant instruction set, chip performance is equal to $IPC * Frequency$. Moore's law [58] predicts that the on-chip transistor count doubles every 18 to 24 months. Its corollary predicts that chip performance will follow a similar law of exponential growth. Both Moore's law and its corollary have been valid for many years. The improvement in chip performance comes from two aspects: the growth of chip frequency and the growth of IPC. Chip frequency has increased at an exponential rate for decades. From generation to generation, technology scaling shrinks the size of semiconductor devices. Scaling not only helps integrate more transistors on chip, but also reduces transistor switching time and therefore improves the processor frequency. Another major method

for improving chip frequency is to use deeper pipelines with their associated enabling techniques. Deeper pipelines need more latches and therefore need more chip area, but until recently this was still an efficient way to utilize chip area and improve frequency. On the other hand, processor performance can also be elevated through IPC growth. A wealth of architectural techniques, such as out-of-order execution, branch prediction, caches etc., have helped improve processor IPC, by supplying instruction level parallelism. In conclusion, past chip designers tended to design complex, wide issue out-of-order processor with deep pipelines and high frequencies to use ever increasing on-chip transistor resources and maintain the processor performance growth predicted by Moore's law.

However, Moore's law can no longer be maintained with only frequency and ILP scaling. First, the performance return from shrinking transistor sizes is diminishing. We need to maintain a relatively high supply voltage to maintain the scaling of transistor switching speed and that will lead to high power density and high chip temperature. Second, superscalar architectures with deeper and wider pipelines are subject to several bottlenecks, including cache misses, branch mispredictions and especially high power consumption. Instead of frequency scaling or ILP scaling, the only way to maintain the rate of performance improvement is to exploit thread level parallelism. There are two basic architectures to exploit thread level parallelism: multiple "cores" in which multiple processors are integrated on one chip and single-core multithreading in which multiple threads share a single processor. Because multithreading incurs contentions for processor resources, multiple cores give much better scalability than multithreading in terms of possible thread count, as this dissertation will show in Chapter 5. Multi-core architectures are taking over monotholic high frequency single core architectures as the mainstream in almost all markets. This dissertation will mainly explore the design space of CMP while also considering the option of using multithreading in CMP architectures, and compare the performance, energy and thermal efficiency of these two techniques.

Further performance improvement not only calls for a transition in architectural methodology, it also calls for awareness of the physical constraints these architectures are facing. As mentioned above, these physical constraints include chip area, chip power consumption, chip temperature and chip pin-bandwidth. In many cases, these constraints turn out to be first-order design optimization

targets. For example, in the mobile and handheld market, power consumption may outweigh performance as the most important design target. The era of performance being the only thing that matters has essentially ended. With technology scaling, many constraints are becoming more severe , such as chip temperature and reliability. These constraints break the traditional micro-architecture abstraction. Micro-architectural performance characterization and architecture design now depend on physical implementation. Ignoring these physical constraints at early design stage is very dangerous. On the other hand, understanding the implication of these constraints on computer architecture is very important because architecture is also a powerful domain to mitigate or exploit physical phenomena.

The CMP design space is vast, especially when future CMP chips may integrate tens of cores. This dissertation serves as the first attempt to explore this design space up to so many cores, and also optimize across so many design variables simultaneously while being aware of all these physical constraints.

The remainder of this chapter introduces the background material for CMP, multithreading and physical constraints, presents the research issues of this dissertation, and lists the major dissertation contributions.

1.1 Background

1.1.1 CMP Background

CMP is an efficient way to utilize chip area by putting more than one core on a single die. The idea of CMP comes from the traditional multi-processor system, especially the symmetric multi-processor system (SMP). While the old style of multi-processor system architecture puts one core on each chip and interconnects them with an off-chip interconnection fabric, CMP integrates multiple cores on a single chip and places the interconnection fabric on the same chip. As mentioned above, CMP has better performance and power scalability compared with complex single core architectures. In addition, CMP has faster core-to-core communication than traditional SMP architectures as well as more flexibility in the memory hierarchy design. For example, several cores in one CMP

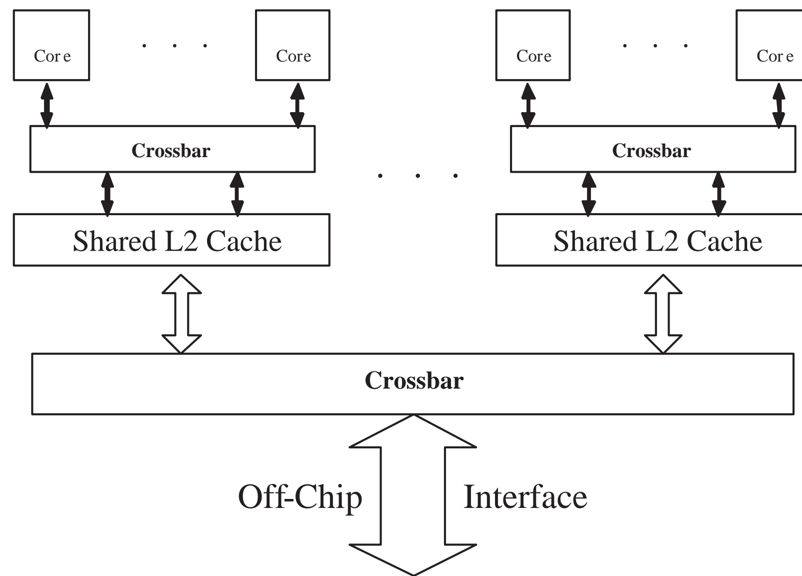


Figure 1.1: **A typical CMP organization**

chip may share an L2 cache which can help improve the efficiency of cache utilization. Figure 1.1 illustrates a typical CMP chip organization. In this figure, several cores share an L2 cache through a crossbar and all these L2 caches connect to the off-chip interface through another crossbar. This off-chip interface is used to transfer traffic from the chip to off-chip memory structures and vice versa.

The CMP design space is huge. The selection of core type is a major design decision for CMP chips. The design dimensions for the type of core includes: in-order (IO) or out-of-order (OO) instruction execution, pipeline depth and width, single-threaded or multi-threaded cores and sizes of storage structures, such as I-cache and D-cache. These decisions are the focus of this dissertation. However, it is important to note that the on-chip interconnection choice is another major CMP design dimension. Possible on-chip interconnection styles include shared bus, crossbar, hierarchical crossbar, mesh, etc. The on-chip cache hierarchy and the cache sharing style is a third major CMP design dimension. Usually each core in a CMP chip has a private L1 cache. L2 caches

Vendor	Chip	Core count	Threads per core	Core type	L2 size (MB)	Cores /L2 cache	Interconnection fabric
IBM	Power4	2	1	OO	1.44	2	Bus
IBM	Power5	2	2	OO	1.92	2	Bus
STI	Cell	9	1-2	Hybrid	0.5	1	Ring
SUN	Niagara	8	4	IO	3	8	Crossbar
INTEL	Core-Duo	2	1	OO	2	2	Bus
NVIDIA	G71	32	N/A	IO	N/A	N/A	N/A
ATI	R580	56	10	IO	N/A	48 (Pixel)	N/A

Table 1.1: **Configurations of commercial CMP chips (OO = Out-of-Order; IO = In-Order)**

can be either shared by several cores or be owned by one single core. These latter issues are beyond the scope of this dissertation. Configurations of some well-known commercial CMP chips are listed in Table 1.1.

As Table 1.1 shows, most commercial CMP chips at the time of this writing integrate two OO cores. This is because this era of CMP design is still in its infancy. At 65nm and 45nm process technology, the problem of how to use transistors on chip will be more serious, leading to much more radical CMP designs with more than eight cores integrated on-chip for server class application. Shrinking transistor size also implies that larger L2 caches can be included on-chip in the future. Increasing core count and L2 cache size for the future CMP chips motivates CMP design space exploration because of the need to answer questions regarding the optimal core count and L2 cache size for a CMP design. Increasing the number of cores also calls for more efficient on-chip interconnection and therefore significantly complicates the CMP design space. Table 1.1 also reveals that most prevalent CMP chips use OO cores. The reason is that OO design has been the prevalent architecture for decades and the industry has tended to reuse core design when migrating to CMP. Although OO chips have better single-thread performance, they are also more complex, larger and more power hungry. Whether a CMP with OO cores is better than a CMP with IO cores under area and power constraints is still an open question, and is explored in Chapter 7. There are three exceptions to the two core standard in this table. The SUN Niagara chip already has 8 cores on chip because it uses IO design for its cores. The SUN Niagara chip targets the server market, which mainly emphasizes the total chip throughput. Another exception is the STI Cell processor. The STI Cell processor targets family entertainment, consumer electronics and gaming markets. It integrates 8 IO cores and 1 OO core on chip. This is an example that CMP can even be heterogeneous [41,42],

consisting of a few complex, powerful cores and many small and simple cores. Heterogeneous CMP designs can accommodate different applications better than homogeneous CMP designs, but usually it also incurs higher verification and design costs. Programming and the operating system scheduling support for a heterogeneous CMP chip tend to be more complex. Heterogeneity will greatly expand the CMP design space and this design dimension is not included in this dissertation. Graphics processors (GPUs), like the NVIDIA G71 chip and the ATI R580 chip shown in this table, tend to integrate large numbers of fairly general-purpose "shaders" (i.e., cores) and the ability to keep many threads in flight. This kind of design stresses throughput over single-thread (single-pixel) latency, and uses the high degree of multithreading to mask memory (chiefly texture) latency.

CMP has better power scalability than single core architectures. However, in the future when there are tens of cores on a chip, the absolute total power required can still be huge and easily exceed the maximum chip power delivery capabilities. Also, high total power consumption will put greater burden on the chip heat removal mechanisms, such as heatsinks and fans. Higher total power consumption will lead to higher chip temperature, (which is discussed in detail in Chapters 6 and 7). So, it is crucial to pay attention to both power and temperature constraints in CMP research. There has been some research work related to CMP design space exploration. Please refer to the related work sections in Chapters 6 and 7.

CMP chips can be used to run either independent programs or parallel applications. Since future CMP chips may contain tens of cores, the problem of providing enough threads for full CMP utilization will be more serious. It may be easy to find hundreds of threads on a web server, but it is very difficult to find enough active threads in a personal computing environment. Virtualization helps, but requires a significant shifting of the computing paradigm. There are two ways to increase thread count. Both can be used to improve single application performance with CMP infrastructures if they can be realized. The first method is to parallelize serial applications. This method puts too much pressure on programmers. It seems that automatic parallelization compilers will not be available in the foreseeable future. It is well known that programming parallel programs is very difficult even with the support of appropriate libraries like PVM and MPI. Some researchers are proposing new programming languages and new coherence schemes to ease parallel programming, but there are

no immediate solutions while there are many legacy sequential programs. The other method is the so called speculative multi-threading (SpMT) technique. Speculative multithreading utilizes otherwise idle hardware thread contexts to execute speculative threads on behalf of the non-speculative thread. These speculative threads can be used to reduce the cache miss ratio, to improve the branch predication accuracy, and/or to speculatively execute the instructions of the non-speculative thread. Run-ahead is one style of SpMT which requires minimal additional hardware support. Mutlu et al. [61] and Dundas et al. [19] proposed this technique. A run-ahead thread is spawned when a cache miss happens or the instruction window is full, and this thread will ignore instruction dependencies and continue to speculatively execute the following instructions while the non-speculative thread is stalling on long latency operations (normally a cache miss). The execution of this speculative run-ahead thread is likely to prefetch further data before it is needed, and therefore can help reduce the cache miss ratio. Compared with traditional prefetching mechanisms, this scheme has higher prefetching accuracy for irregular memory access patterns because it executes real instructions in advance. Sun Rock processor's "scouting" mechanism [13] is an example of this technique. Other than this run-ahead technique, there are many other SpMT techniques which have been proposed. They normally require support from the compiler and their hardware implementation is more complex than the run-ahead method. Good examples of these kinds of techniques can be found in [34] and [56]. Although the problem of how to provide enough threads for future CMP chips is an extremely important problem in CMP research, it is outside the scope of this dissertation. This work assumes that at any time there are enough threads to fill all cores on chip. This state will be achieved if this thread count problem is solved in the future. It also represents the heavy load scenario for current CMP systems and in the near future.

1.1.2 Multithreading Background

There are at least two different multithreading implementations: coarse-grain multithreading, and simultaneous multithreading. Coarse-grain multithreading is designed to take advantage of otherwise idle cycles incurred from a long-latency event like a cache miss leading to off-chip memory access. When such an event happens, with a coarse-grain multithreading support, another thread

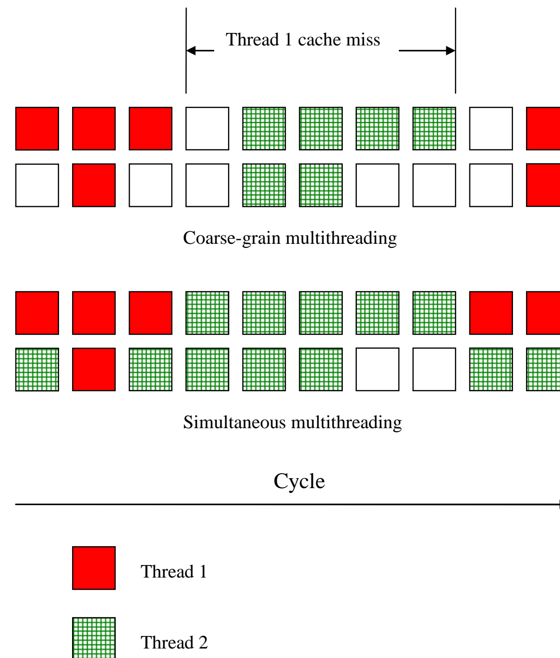


Figure 1.2: **Coarse-grain multithreading vs. simultaneous multithreading**

can be scheduled to execute. The execution of this thread will overlap with long-latency event cycles and improve the system throughput. Coarse-grain threading is used on the IBM pSeries* S85 [5]. Similar techniques are also investigated in J. Haskin's work on dMT [27].

Simultaneous multithreading (SMT) [76] can schedule instructions from all threads in the same cycle. Ideal resource allocation to each thread depends on the requirements of each thread and the balance among them for fairness. In reality, the scheduling algorithm must be hardware affordable. Therefore simple scheduling algorithms , including round-robin and ICOUNT [75], were proposed to approximately achieve the efficiency and the fairness of resource utilization in SMT architectures.

Figure 1.2 shows the basic difference of these two multithreading mechanisms. It assumes an architecture with two pipelines. The top figure illustrates coarse-grain multithreading while the bottom figure illustrates SMT. In the coarse-grain multithreading figure, when thread 1 suffers a cache miss, thread 2 can be scheduled and swapped in to take advantage of the idle pipelines, but in this case thread 1 and thread 2 never share any pipelines simultaneously. However, for the case of SMT shown by the figure at the bottom, thread 1 and thread 2 can share pipelines all the time. Therefore threads in coarse-grain multithreading only share pipelines vertically (or in time) while threads in SMT share pipelines both vertically and horizontally (or in both time and space).

Threads in coarse-grain multithreading tend to share more resources than in SMT. For example, in coarse-grain multithreading architectures, threads normally share the architecture register file, therefore the thread switch has quite high overhead. This overhead can be tolerated in coarse-grain multithreading architectures because coarse-grain multithreading is designed to hide very long latency events. But this kind of overhead cannot be tolerated in SMT architectures because of their finer granularity for thread switching. With increased resource duplication and scaling, SMT can achieve more throughput improvement than coarse-grain multithreading. The commercial applications of SMT include Intel's Pentium4 with hyperthreading and IBM's Power5. For detailed introduction of related work for SMT research, please refer to the Related Work section in Chapters 3 and 5. There are other multithreading implementations like fine-grain multithreading which is used in the Tera computer system [2]. Their implementation philosophy usually sits between the two extremes of coarse-grain multithreading and SMT. For example, fine-grain multithreading can hide events with shorter latency than a memory access, but its threads are executed in a round-robin fashion, therefore it cannot simultaneously execute multiple threads.

1.1.3 Physical Constraints

This dissertation work considers four major physical constraints. They are chip power, chip temperature, chip area and chip pin-bandwidth. Chip power consumption is not only a constraint but also a major design optimization target in many cases. Power consumption affects system behavior in a number of ways: First, for handheld devices it largely determines system battery life. Second,

Year	2006	2010	2013	2016
Tech node(nm)	70	45	32	22
Vdd (high perf)(V)	1.1	1.0	0.9	0.8
Frequency (high perf) (Ghz)	6.783	12.369	22.98	39.683
Max power (high perf) (W)	180	198	198	198
Size (mm^2)	195	310	310	310

Table 1.2: **ITRS roadmap**

for server systems high power consumption and the associated data center cooling costs lead to high utility expenses. Third, power consumption and power density distribution on the chip determines chip temperature and high chip temperature usually leads to high package cost. Finally, power delivery competes with off-chip traffic for pin-bandwidth while chip pins are already a limited resource. Since chip area is not shrinking from generation to generation and power density increases with technology, total chip power is also increasing. Therefore this constraint is becoming more and more serious. From the 2005 ITRS prediction listed in Table 1.2, we can see that the maximum power estimation is fixed at 198W for the 45nm, 32nm and 22nm technology. But in 2001 ITRS prediction, the maximum power estimation for 22nm technology was 288W. This reduction shows that such high power delivery will not be affordable in the future, especially due to the power density problem. Therefore any future CMP research must take the power constraint into consideration.

Power consumption includes two parts: The dynamic power consumption from transistor switching and the leakage power consumption. Leakage power is also exponentially dependent on temperature. For a typical chip working at 373K, the leakage power can constitute more than 40% of the total power consumption in 65nm technology. This ratio is only around 10% for 130nm technology. Because leakage power plays a more significant role in total power consumption at future technology, this dissertation models this factor.

Because chip power density is increasing with technology, the constraint due to chip temperature is also becoming more severe. High chip temperature not only leads to high packaging costs, but also leads to worse life-time reliability because many failure mechanisms (like electromigration) are exponentially dependent on the chip temperature. For all of these reasons, chip

temperature is also becoming a major design constraint.

Chip area is also a major constraint in designing a CMP. Bigger chip area means smaller yield and therefore increased chip manufacturing cost. Table 1.2 shows that the chip area will not increase. It remains constant for 45nm, 32nm and 22nm in ITRS 2005 prediction. Due to different chip pricing for different markets, the specific chip area constraints for different markets are different. Generally high end server chip tends to be around 400 mm^2 while desktop and laptop chips are around 100 m^2. In a CMP design, a bigger chip can hold more cores and may provide higher throughput, therefore the constraint of chip area is crucial for CMP designs.

Finally, this dissertation also considers the chip pin-bandwidth constraint in Chapter 7. Pin-bandwidth limitations are likely to be an increasing challenge for designers of multi-core CPUs as future CMP chips will integrate tens of cores and each core can be multi-threaded. Using optical links can increase chip pin-bandwidth greatly, but this technique is still expensive at this time. Integrating EDRAM on chip can help reduce the off-chip bandwidth, but this technique is still not widely available in industry. Currently, chip pin-bandwidth is limited by the number of chip pins and on-chip DDR/DRAM controllers. Nowadays a typical server chip like the STI Cell chip and the Sun Niagara chip can provide 25GB/s or so pin-bandwidth.

In reality, all these constraints are inter-related. For example, when the area constraint is relaxed, a better CMP design in terms of throughput tends to put more cores on chip; however, if there is a temperature constraint, it is not always beneficial to put as many cores as possible on chip, because the performance benefits from adding more cores may be negated by the extra thermal stress and requisite throttling introduced by these cores. Another example is that more pin-bandwidth requires more on-chip DDR/DRAM controllers, which consumes more chip area. More pin-bandwidth may also require more chip pins, which will decrease the number of pins used to deliver chip power and therefore decrease chip power delivery capabilities.

1.2 Research Issues and Dissertation Overview

This dissertation first builds the models necessary to simulate the performance, power, area, temperature and chip pin-bandwidth of both CMP and SMT architectures. It then uses these models to explore the vast design space of CMP subject to the various physical constraints and shows the fundamental tradeoffs in future CMP designs. The main research issues and contributions are summarized as follows.

- What are the best CMP configurations under physical constraints. Chapters 6 and 7 are dedicated to this topic. This research optimizes the following design dimensions: core type (in-order vs. out-of-order, single-thread vs. SMT), core number, pipeline depth, pipeline width and L2 cache size. Because the design dimension of the core type is of special interest to the architecture community, it is discussed in detail in Chapter 7 separately.

- Chapter 6 explores the multi-dimensional design space for chip multiprocessors, investigating the inter-related variables of core count, pipeline depth, superscalar width, L2 cache size, and operating voltage and frequency, under various area and thermal constraints. The results show the importance of joint optimization. Severe thermal constraints can dominate other physical constraints such as pin-bandwidth and power delivery, demonstrating the importance of considering thermal constraints while optimizing these other parameters. For aggressive cooling solutions, reducing power density is at least as important as reducing total power, while for low-cost cooling solutions, reducing total power consumption is more important. This work is also published in [53].

- Chapter 7 investigates a wide spectrum of core types in CMP designs, including in-order vs. out-of-order, non-SMT vs. SMT, different level one cache sizes and different pipeline depths and widths. Designers of future chip-multiprocessor systems will be increasingly required to optimize for a combination of single-thread performance, total chip throughput, and energy. This chapter explains that for systems that must meet single-thread latency targets, aggressive out-of-order cores can provide superior single-thread latency in addition to competitive

or in many cases superior aggregate throughput. This conclusion holds even when latency constraints are relaxed to 10x of a conventional OO architecture. Chapter 7 also finds that in-order cores do provide some benefits from a thermal standpoint; when thermal constraints are quite severe, in-order cores can be superior when latency constraints are small and throughput is of primary concern.

- It requires significant modeling efforts to build up an infrastructure that is capable of this CMP design space exploration subject to physical constraints. The basic performance, power and temperature modeling methodology is introduced in chapter 2. The methodology of power and area scaling with different pipeline depths and widths is presented in chapter 6. Chapter 7 introduces the modeling of an in-order architecture and the modeling of chip pin-bandwidth.

- CMP simulation is very costly. The CMP design space search in this dissertation requires numerous CMP simulations. Brute force simulation is unaffordable for this task. A CMP simulation acceleration technique - Zauber - is proposed in this dissertation to help the CMP design space exploration. This method decouples the core simulation and the interconnection/L2 cache simulation and improves the CMP simulation speed by more than 100 times while retaining good accuracy. This approach is introduced in detail in chapter 6. It is also published in [53].

- Compared with single-threaded architectures, SMT architectures need to scale many resources to avoid excessive resource competition and provide the optimal performance. This optimal scaling factor is important for this CMP research and the comparison of CMP and SMT. Chapter 3 introduces modeling extensions to an architectural simulator which allow the study for the power-performance efficiency of SMT. This modeling extension is used to do a thorough research on SMT power efficiency analysis and give the optimal scaling factor for the best power-performance tradeoff for SMT. This serves as the foundation for the work in chapter 5 and 7. This work is also published in [52].

- Clock gating is a major technique for saving power. Different clock gating styles can change the chip power density and temperature dramatically. It is therefore important to investigate

the efficiency of different clock gating styles for different units on chip and their interactions with architectural factors. This research is presented in chapter 4. It is also published in [51].

- The comparison of CMP and SMT architectures. It is important to understand the fundamental difference in terms of performance, power and thermal efficiency between these two architectures before we try to consider incorporating SMT in CMP. Chapter 5 does an equal area comparison of a two threads SMT chip and a dual-core CMP chip. The results show that CMP is superior in terms of performance and energy-efficiency for CPU-bound benchmarks, but SMT to be superior for memory-bound benchmarks due to a larger L2 cache. Chapter 5 also shows the fundamental difference in heat up mechanisms for these two architectures.

Chapter 2

Experimental Methodology

2.1 Simulator

2.1.1 Baseline Model

The detailed core simulation infrastructure used in this dissertation work consists of Turandot, PowerTimer, and HotSpot 2.0. Turandot is a validated model of an IBM POWER4-like architecture [60]. PowerTimer implements circuit-extracted, validated power models, which has been extended with analytical scaling formulas based on Wattch [6, 8] in this work. HotSpot 2.0 is a validated, architectural model of localized, on-chip temperatures [70]. Each of these components in this detailed simulation infrastructure is modular so that any particular simulator can be replaced with an alternative.

2.1.1.1 Baseline Performance Model

Turandot models an out-of-order, superscalar processor with resource configuration similar to current generation microprocessors. The overall processor organization is shown in Figure 2.1 and Table 2.1 describes the configuration of the baseline processor for the single-threaded design point. As shown in Figure 2.1, the simulated processor can be logically divided into six major units: IFU, IDU, ISU, LSU, FXU, and FPU. The components of these units are listed below:

- Instruction Fetch Unit (IFU): IFU includes program counters, level-one instruction cache, instruction TLBs, instruction buffer, branch predictor, next fetch address predictor (NFA), return address stack, etc.

- Instruction Decode Unit (IDU): IDU includes instruction decoder, microcode ROM, etc.

- Instruction Sequencing Unit (ISU): ISU includes register renamers, reservation stations, and retirement queue, etc.

- Load/Store Unit (LSU): LSU includes effective address calculator, level-one data cache, data TLBs, cast-out queue, load reorder buffer, store queue, load miss queue, etc.

- Fixed-point Execution Unit (FXU): FXU includes integer ALUs, integer multipliers/dividers, shifters, integer register file, etc.

- Floating-point Execution Unit (FPU): FPU includes floating-point pipelines, floating-point register file, etc.

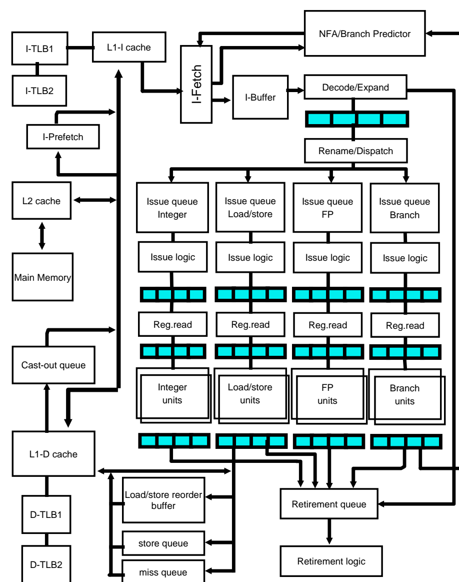


Figure 2.1: **Modeled Processor Organization [52]**

Processor Core	
Dispatch Rate	5 instructions per cycle
Reservation stations	mem/fix queue (2x20), fpq (2x5)
Functional Units	2 FXU, 2 FPU, 2 LSU, 1 BRU
Physical registers	80 GPR, 72 FPR
Branch predictor	16K-entry bimodal, 16K-entry gshare, 16K-entry selector, all with 1-bit entries
Memory Hierarchy	
L1 Dcache Size	32KB, 2-way, 128B blocks
L1 Icache Size	64KB, 1-way, 128B blocks
L2 I/D	1MB, 4-way LRU, 128B blocks 9-cycle latency
L3 Cache/Memory Latency	77 cycles

Table 2.1: **Configuration of simulated processor**

2.1.1.2 Baseline Power Model

PowerTimer differs from existing academic microarchitectural power-performance simulators primarily in energy-model formation [9, 29]. The base energy-models are derived from circuit-level power analysis that has been performed on structures in a current, high-performance PowerPC processor. This analysis has been performed at the macro level, and in general, multiple macros will combine to form a microarchitectural level structure corresponding to units within the performance model. PowerTimer models over 60 microarchitectural structures which are defined by over 400 macro-level power equations. Unless explicitly mentioned, I assume uniform leakage power density for all the units on the chip if they have the same temperature. Leakage power is estimated based on a formula derived by curve fitting with the ITRS data [68]. Leakage power of one unit depends on the area and temperature of that unit. Incorporating more accurate leakage power models will improve the accuracy of the results, especially for future technologies—an important area for future work.

These energy models are tightly coupled with Turandot, the performance simulator described in Section 2.1.1.1. The unconstrained power estimates are then scaled by microarchitectural utilization information to estimate clock-gated power dissipation. The remainder of this section focuses on issues that will specifically be impacted by simultaneous multithreading and dynamic thermal management: in particular, the clock gating and resource scaling methodologies.

PowerTimer uses microarchitectural activity information from the Turandot model to scale down the unconstrained power under a variety of clock gating assumptions. This study uses a realistic form of clock gating which considers the applicability of clock gating on a per-macro basis to scale down the power depending on microarchitectural event counts. This study determines which macros can be clock gated in a fine-grained manner (per-entry or per-stage clock gating) and which can be clock gated in a coarse-grained manner (the entire unit must be idle to be clock gated). For some macros (in particular control logic), no clock gating is applied; this corresponds to about 20-25% of the unconstrained power dissipation. Typically, the overall savings due to clock gating relative to the unconstrained power is roughly 40-50%. SMT machines tend to increase the utilization of the pipeline, diminishing power reduced by clock gating.

There are several styles of clock gating that are applied depending on the specific macro. These include valid and stall gating for latch-based structures and read and write port gating for array structures. Valid-bit clock gating is commonly used in pipeline latches and relatively small memory structures that are designed using latch-and-mux schemes (e.g. issue queues, instruction buffers, etc). In this style of gating, a valid-bit is associated with every bank of latches and the local clock buffer of the latch bank is gated when the valid-bit is not set. For array structures such as caches and large RAM-banks in certain queue structures, the array structure utilization is proportional to the number of read and write accesses to the structure. This dissertation introduces and compares different clock gating choices for array structures in Chapter 4.

2.1.1.3 Baseline Temperature Model

To model operating temperature, this study uses HotSpot 2.0 (http://lava.cs.virginia.edu/HotSpot), which accounts for the important effects of the thermal interface material (TIM) between the die and heat spreader and has been validated against a test chip [30].

HotSpot models temperature using a circuit of thermal resistances and capacitances that are derived from the layout of microarchitecture units. The thermal package that is modeled consists of the die-to-spreader TIM (thickness 0.05mm), the heat spreader (thickness 1mm), another TIM, the heat sink (thickness 6.9mm), and a fan. Removal of heat from the package via airflow takes place

by convection and is modeled using a single, equivalent thermal resistance . This assumes the fan speed and the ambient temperature inside the computer "box" (40°C) are constant, both of which are true for the time scales over which all benchmarks are simulated.

Due to lateral heat spreading, thermal behavior is sensitive to the layout of the microarchitecture units. This study uses the floorplans shown in Figure 2.2, which have been derived by inspection from the die photo of the POWER5 in [14]. Note that Figure 2.2 only shows floorplans for the single-threaded and CMP chips. The SMT floorplan is identical to the single-threaded case, except that the increase in resources to accommodate SMT makes the core 12% larger. (This is small enough—a few percent of the total chip area—that the impact on L2 size for SMT can be taken as negligible.)

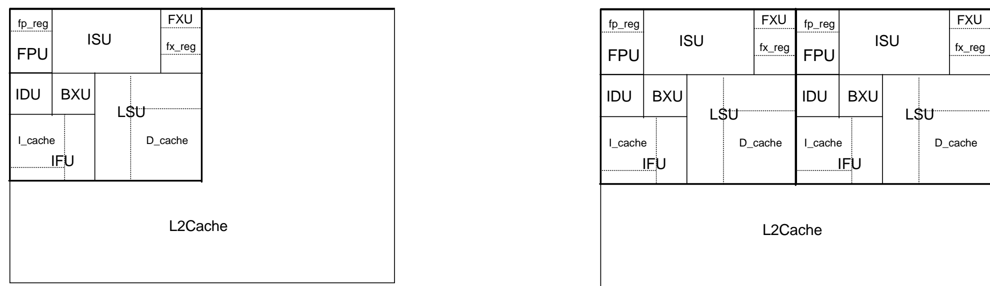


Figure 2.2: **Floorplans used for thermal simulation. The SMT core is 12% larger than the ST core shown above. [54]**

2.1.2 SMT Extension to the baseline Model

This section describes the extensions that are added to the Turandot/Powertimer model for supporting SMT. A distinguishing feature of SMT is that execution units (FXU, FPU, etc) are usually shared among threads, thread-specific resources, such as program counters, are always duplicated, while the rest of resources (branch predictor, caches, etc) can either be shared or duplicated depending on design choices. Since all these resources are already modeled in the single-threaded base model, the extensions to SMT are straightforward. In addition to resource extensions, extra control logic is needed at various pipeline stages to decide which threads should go ahead, while others should be stalled on a given cycle. A simple policy that is commonly used is "round-robin", where the choice of the target thread is rotated sequentially (with wrap-around) among the available

threads. This is the default thread selection policy implemented in the new SMT-enabled performance model. In future work, more sophisticated thread prioritization policies will be added and tested.

SMT impacts the resource utilization within a microprocessor. This impact will vary depending on the style of clock gating that exists in the underlying structures. For example, if the occupancy rate of queues increases, structures with valid-bit based clock gating are likely to see increases in power dissipation. On the other hand, the impact of SMT on array structures may be small if the total number of accesses is roughly constant. For example, for the same workload, one would expect roughly the same number of loads and stores with SMT and non-SMT architecture, as this is primarily dependent on the trace being executed, not the microarchitecture.

The majority of structures in a superscalar pipeline can be shared when augmenting the microprocessor for SMT. However, architected state must be duplicated for additional threads and new performance bottlenecks may arise requiring extension of shared resources. The major anticipated resource needs for SMT extensions can be categorized into the following.

- Resource Duplication. Structures such as the program counter must be duplicated for each thread. In this case power dissipation has been increased proportionally to the number of threads in the machine.

- Latch-Based Queue Structures. With latch-based queue structures, the power dissipation is dominated by the latch and clocking circuitry, and increases nearly linearly with the increase in the number of entries and bits per entry of these structures. The following formula is used:

$$\mathrm{P}ower_{new} = \frac{Entries_{new}}{Entries_{base}} * Power_{base} * PowerFactor \tag{2.1}$$

  The default power model assumes that *PowerFactor* is 1.0 (linear scaling). Li et al. [52] showed that the choice of power factor does not change relative behavior.

- Array Based Structures. For array-based structures, the empirical macro-level data from PowerTimer has been used for a base value, and this base value is then scaled for size and

associativity with estimates based on analytical models built into Wattch [8].

2.1.3 CMP Extension to the Baseline Model

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

2.2 Benchmark

SPEC2000 benchmarks are used for the research work in Chapters 3, 4 , 5 and 6. This dissertation also uses SPECJBB in Chapter 7. Detailed sampling and tracing methodology is introduced in these chapters. Single-thread benchmarks are combined to form SMT/CMP benchmarks. The methodology of combining single-thread benchmarks to form SMT/CMP benchmarks is presented in Chapters 3, 5, 6 and 7.

2.3 Speedup Metric

Comparison of different SMT configurations, or comparison of an SMT configuration against a single-threaded configuration, is difficult. As Sazeides and Juan [65] have shown, IPC can be misleading unless exactly the same instruction count for each thread is used in all experiments. Otherwise, a high IPC may be achieved with a skewed load balance. Snavely et al. [71] also argue that SMT simulations should not be stopped when the first thread completes to perform comparison only on the portion of a workload that experiences multithreaded execution. This unfairly benefits SMT configurations by not accounting for periods of less than maximum throughput. When performing energy-efficiency studies, it also overlooks the impact of SMT energy overheads that are present even when only one thread is executing. Both groups propose similar metrics for computing an “SMT speedup”. The goal is to distinguish between configurations that achieve high throughput at the expense of a single thread from those that do so with balanced throughput from both threads.

Sazeides and Juan propose that

$$\text{SMT speedup} = \frac{\sum L_{nonSMT}[i]}{L_{SMT}} \tag{2.2}$$

where L_i is the execution latency of the i'th thread on a single-threaded system, and L is the execution latency of the workload on an SMT system. A drawback to this mechanism is that L_{SMT} is determined by the thread that finishes last, and it cannot distinguish between different execution rates for other threads.

Snavely et al. propose that

$$\text{SMT speedup} = \sum \frac{IPC_{SMT}[i]}{IPC_{nonSMT}[i]} \tag{2.3}$$

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

In contrast to evaluating performance, evaluating energy efficiency should use traditional, simple unweighted metrics. Total energy consumed during an experiment is the appropriate value to use for energy metrics, and simple end-to-end execution latency is the appropriate value to use for delay with energy-efficiency metrics like energy-delay^2. There are two reasons for this. First, unlike the tradeoff between energy and execution speed, it is not clear how to trade off energy and load balance. Second, using weighted SMT speedup in an energy-efficiency metric could yield the counter-intuitive result that, among two SMT configurations with equal end-to-end execution latencies, a result with higher energy consumption is preferred. The argument in this section also applies to CMP. Therefore the same methodology is used for CMP speedup and energy consumption calculation.

Chapter 3

The Energy Efficiency of Simultaneous Multithreading

3.1 Introduction

Simultaneous multithreading (SMT) [76] is a relatively new microarchitectural paradigm that has found its way into real products [35,55]. The promise of SMT is area-efficient throughput enhancement; however, the significant boost (10-40%) in instructions per cycle (IPC) is accompanied by an increase in power consumption. Since the area increase reported for SMT execution is relatively small (less than 5% per chip) [16], the main concern in next-generation SMT processor design is that of worst-case power and temperature characteristics.

There has been recent related work in understanding area efficiency and power issues in multithreaded processors. Burns and Gaudiot [11] consider the scalability of various resources in an SMT processor and perform a detailed study of the area overhead of SMT processors. Seng and Tullsen study several power-aware optimizations in the context of a fixed-resource multithreaded microprocessor [66]. This work recognizes that increased processor utilization in SMT machines will impact power, but not area, and focuses on understanding the fundamental power-performance efficiency of SMT rather than SMT-specific power optimizations.

This work provides several major contributions. First, a thorough design space exploration is provided to understand the performance benefits and power costs of SMT in the context of extensions to an existing POWER4-like microarchitecture. This exploration shows that SMT is a very power-efficient design paradigm in terms of ED^2 and can provide a 20% performance improvement

for a varied mix of workloads with a power overhead of around 24%. There are several underlying reasons for power uplift (the same as power increase) in SMT machines, and the uplift is diagnosed by analyzing the machine at the unit level in this work. The impact of future technologies is also analyzed where static leakage power is more significant, and it can be determined that the power overhead of SMT decreases with leakier process technologies because the power uplift due to utilization is marginalized by the larger fraction of leakage power. Finally, this chapter discusses the sensitivity of conclusions to modeling assumptions. This work is also published in [52].

3.2 Benchmark Methodology

For this study, 10 SPEC2000 integer benchmarks are used for single thread experiments. They are compiled by *xlc* compiler with -O3 option. The static trace generation tool generates the final static traces by skipping the first 1B instructions and then tracing for 100M instructions in 50M instruction chunks, skipping 100M instructions between chunks.

Pairs of single-thread benchmarks are used to form dual-thread SMT benchmarks. There are many possibilities for forming the pairs from these 10 benchmarks. The following methodology is used to form pairs. First, each single thread benchmark combines with itself to form a pair, which gives a total of 10 pairs. Then several pairs are formed by combining different benchmarks, after categorizing the benchmarks into four major categories: high IPC or low IPC, memory intensive or not memory intensive. Six pairs are finally formed (gzip+perlbmk, gcc+gap, twolf+mcf, parser+bzip2, bzip2+twolf, gcc+mcf) of dual-thread benchmarks by selecting unique combinations of benchmarks with these categorizing criteria.

3.3 Results

This section discusses the relative power-performance efficiency of SMT, analyze the relative impact of SMT power uplift factors, and discuss sensitivities to resource sizes, leakage power, and the power modeling methodology.

3.3.1 Power-performance Efficiency of SMT

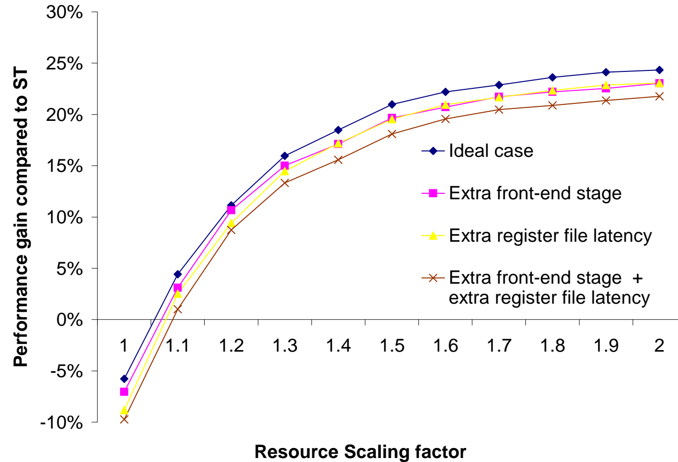


Figure 3.1: **Performance of SMT vs. ST [52]**

To provide a balanced approach to support the increased number of inflight instructions provided by SMT, a "ganged" scaling is provided of all instruction buffers/queues, including instruction buffer, retirement queue, reservation stations, and physical registers. This chapter uses *resource scaling factor* to indicate the magnitude increase of queues, buffers, and physical register files compared to the base case shown in Table 2.1. A resource scaling factor of 1.0 corresponds to the case where these structures are sized the same as the base case, while a resource scaling factor of 2.0 means all the structures mentioned are double-sized. The memory hierarchy and memory-related queues (including load reorder queue, store reorder queue, and load miss queue) are not scaled, because the sensitivity study indicates that their current sizes, as in the base case, do not constitute a performance bottleneck for the benchmarks this chapter studies. Since only part of the resources are upscaled, the overall core area increase (excluding L2 cache) is estimated to be around 10% with a resource scaling factor of 1.5.

Figure 3.1 summarizes the performance benefit for SMT over the baseline single-threaded (ST) microprocessor, when varying the resource scaling factor. The numbers shown are the average performance for all the SMT pairs that are simulated. The four curves correspond to different assumptions about the extra latencies SMT will incur, including the ideal SMT machine, where no

extra latencies are added, a machine with an added pipestage in the front-end to account for thread selection logic, a machine with an added pipestage in the register file access to account for the larger register file, and finally, a machine that incurs both of the above latencies. We see from the figure that, at resource scaling factor of 1.0, thread contention is a serious problem, and even the ideal SMT machine suffers a 5% performance loss. The performance increases significantly when extra resources are added. At 1.5x scaling, the SMT performance benefit increases to around 21% for the ideal machine and 19% for the machine with both additional latencies. The curves begin to saturate with resource scaling factor of about 1.5, after which point increasing resources sees only diminishing performance gains. An interesting observation from the figures is that the four curves have very similar trends, indicating that the different latency assumptions do not change the SMT performance trend while varying resource scaling factor.

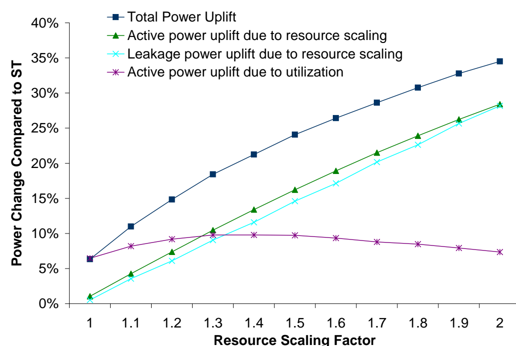


Figure 3.2: **Power Dissipation of SMT vs. ST [52]**

Now we look into the power dissipation of SMT. There are two major factors that cause SMT power uplift – the uplift due to resource duplication and resource sizing and the power uplift due to increased utilization (leading to reduced clock gating potential). PowerTimer allows me to measure the contributions of these two major components by providing power statistics with and without the power uplift applied by resource scaling. Figure 3.2 details the additional power dissipation that SMT incurs over the single-threaded machine and breaks down the two components of SMT power uplift. At the 1.5x scaling point the total core power has increased by 24% relative to the single-thread machine. The increase in processor utilization accounts for about 8% of this power increase and the remainder is due to the increased resource sizings. The power uplift due to processor

utilization exhibits an interesting trend – with very small (1x) and very large (2x) values of resource scaling factor the power uplift is relatively small (5-6%). This trend is explored in more detail in Section 3.3.2 when the per-unit power uplift breakdown is discussed.

Figure 3.2 also breaks down the power uplift due to increased leakage power as the size of resources is increased. In the model this work uses, leakage power is estimated as a fraction of *unconstrained* active power and this work does not scale leakage power with utilization. For the baseline model, this fraction is assumed to be 0.1 and this variable is called as *leakage factor*. We see that for the power uplift due to resource scaling, leakage power and active power track very closely. However, because the leakage power does not incur the additional power overhead due to increased utilization in the SMT machine, leakage power does not grow as quickly as active power. Section 3.3.3 considers the sensitivity of results to leakage factor.

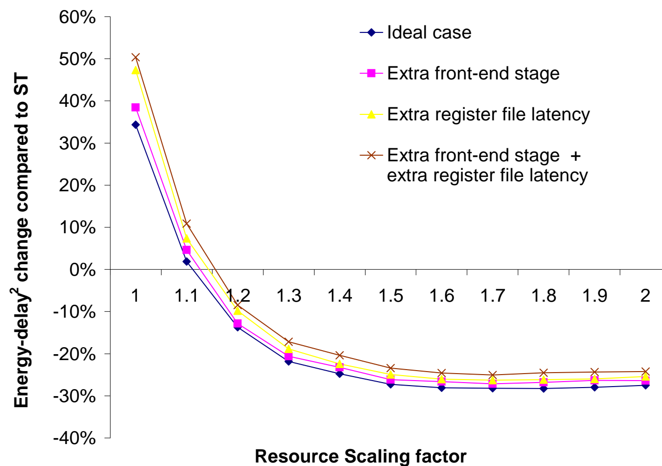


Figure 3.3: **Energy-Delay^2 of SMT vs. ST [52]**

For high-performance processors power-efficiency can best be quantified by the Energy-Delay^2 metric. Improvements in ED^2 correspond to power-performance efficiency benefits that exceed the cubic benefit derived by simply tuning the full-chip clock frequency and supply voltage [7]. Figure 3.3 provides the results for ED^2 and we can see that SMT is indeed very power-efficient and ED^2 is minimized with the SMT processor with 1.6x resource scaling. This is not surprising, given that SMT performance gain starts to saturate at around 1.5x, while power dissipation increases continue with larger values of resource scaling factor.

3.3.2 Breakdown of SMT power overheads by unit

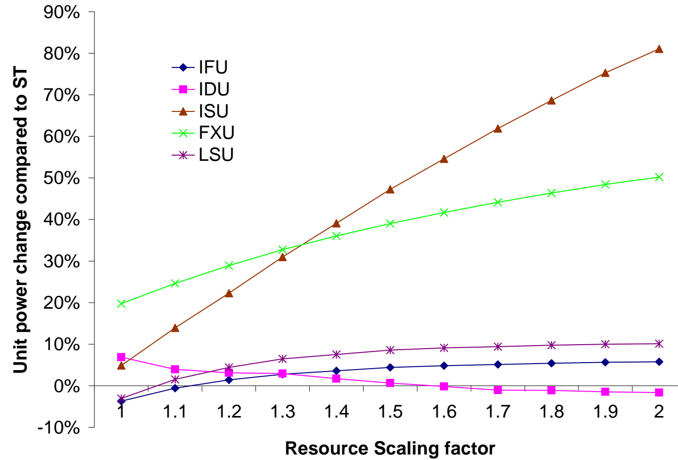


Figure 3.4: **Power Dissipation Breakdown by Units [52]**

We can obtain a better understanding of the power overheads associated with SMT by breaking down the power uplift by unit. Figure 3.4 shows the power increase under SMT for five major units within the microprocessor. The instruction sequencing unit (ISU) clearly stands out as experiencing the largest power changes, primarily because almost all of its subunits, such as reservation stations, register renamers, and retirement queue, are scaled to support SMT. The fixed-point execution unit (FXU) exhibits similar behavior, albeit milder, because the integer register file, which is also scaled under SMT, is in this unit. On the other hand, the power dissipation increase in the instruction fetch unit (IFU) and load/store unit (LSU) is primarily a result of increased utilization, as most of their components stay unchanged from ST to SMT. With more load/stores executed under SMT, at 1.5x resource scaling the LSU dissipates 10% additional power. The utilization uplift of both of these structures saturates when the larger instruction buffers and register files become large enough to support the ILP in both simultaneous threads.

The instruction decoding unit (IDU) displays behavior that is quite different than the other units. At the 1x scaling ratio, the power increase with SMT is roughly 10%, but this power delta gradually reduces as the resources increase. The investigation reveals that at 1x scaling, the small instruction queues and physical register files are a severe performance bottleneck. This causes congestion

within the IDU since the IDU decouples the IFU and ISU. Since the IDU utilizes valid-bit based clock-gating, the increased occupancy leads to higher power dissipation compared to the single-thread base case. As the resources in the ISU is upscaled, the performance bottleneck at the ISU is gradually removed, reducing IDU power.

3.3.3 Sensitivity to Leakage Power

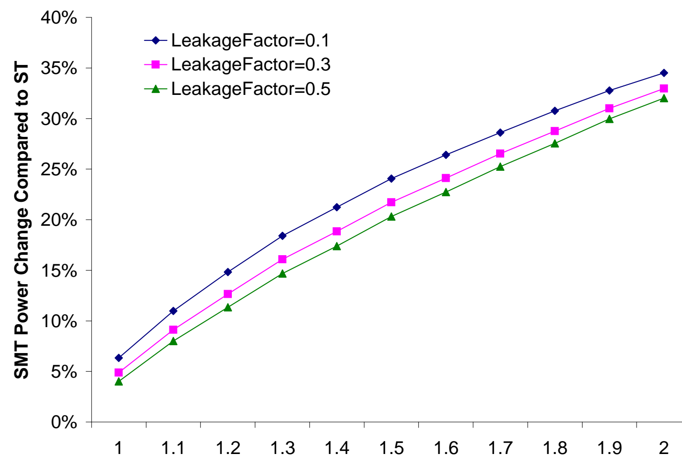


Figure 3.5: **Impact of Leakage on Power Dissipation of SMT vs. ST [52]**

As process technologies migrate to smaller channel lengths and lower threshold voltages, static leakage currents become a major concern in the design of microprocessors. This section considers the power overheads of SMT compared to single-thread architectures in technologies where leakage is a significant fraction of total power dissipation.

Figure 3.5 shows the total power increase (including active and leakage) of SMT compared to the single-thread baseline machine. The future technologies is represented by varying the leakage factor (LF) of the design from 0.1 (the baseline) to 0.5. As described in Section 3.3.1, leakage factor is defined to be the fraction of the total unconstrained chip power that is leakage power.

Figure 3.5 shows that the total power uplift decreases slightly with leakier process technologies. At the 1.5x scaling point, the total power uplift is 24% with LF = 0.1, but reduces to 20% with LF = 0.5. This result is intuitive – as active power becomes a smaller fraction of the total power

dissipation, the SMT machine's increase in utilization, and the corresponding reduction in clock gating potential, has less of an impact because clock gating only reduces active power.

3.3.4 Sensitivity to Resource Power Scaling

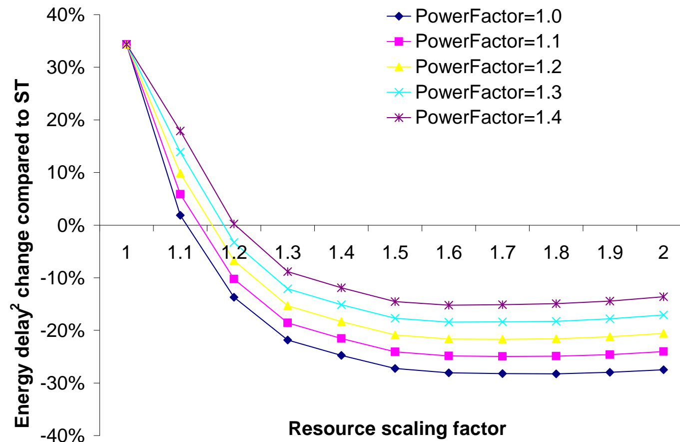


Figure 3.6: **Impact of PowerFactor on Energy-Delay^2 of SMT vs. ST [52]**

This section considers the impact of the scaling assumptions on estimates for ED^2. For many of the structures, as resources are scaled, it is assumed that an increase in the number of entries has a linear increase in the unconstrained power dissipation, i.e. $PowerFactor = 1.0$ in Equation 2.1.

However, there may be cases where this assumption is too conservative. For example, the input and output (de)-multiplexors may become a more significant portion of power dissipation for large queue structures, and this could cause the power to grow super-linearly. Figure 3.6 shows the impact of varying the $PowerFactor$ variable in Equation 2.1 from 1x (linear scaling) to 1.4x. It is apparent that while the overall ED^2 savings will decrease considerably, the optimal design point is around 1.6x for all the $PowerFactor$ this work considers. This indicates that a power model with a slightly inaccurate $PowerFactor$ can still have a meaningful projections for the trend for ED^2 of the SMT processor. This is encouraging, since for many architectural studies, relative accuracy is sufficient because early-stage architectural studies are primarily intended to narrow the focus of design choices. Later studies after design implementation begins can provide more detailed models that improve absolute accuracy.

3.4 Future Work and Conclusions

The future work can seek to further validate the performance and power extensions for SMT. The baseline single-threaded performance model has been extensively validated against a pre-RTL, latch-accurate processor model for a current generation microprocessor [59]. For the SMT extension, the future work can focus on validating the two major perturbations caused by SMT: increased utilization and resource scaling. The strategy to validate utilization is as follows: Simple microbenchmarks can be constructed, which are mainly loop-like kernels, whose resource utilization can easily be deduced under single-thread or SMT environment. Your can run such microbenchmarks, collect the utilization, and compare them with the offline calculation to make sure they match each other. Another possibility is to validate the SMT performance model against the product-level processor model for the most recent IBM processor.

It is also very useful to analyze the worst-case temperature behavior of key structures within the microprocessor, in particular within the ISU, which incurs a sharp increase in power dissipation with SMT.

This chapter describes the modeling extension and validation strategy to study the power-performance impact of SMT. It also performs a detailed design space study of the impact of augmenting an existing POWER4-like microarchitecture with SMT. In conclusion, SMT is a power-efficient design paradigm for modern, superscalar microarchitectures. After careful resource size tuning within the processor core, designers can expect performance gains of nearly 20% with a power uplift of roughly 24% leading to significant reduction in ED^2.

Chapter 4

Power and Energy Efficiency of Different Clock Gating Styles

4.1 Introduction

As modern CPU designs face power and thermal bottlenecks, designers typically adopt *clock gating*—gating off the clock signal to unneeded units, thus reducing dynamic power dissipation. Although there has been quite a lot of circuit-level work on clock gating [22,73], there is very little work from the perspective of architecture. Brooks et al. [8] describe how to model clock gating in an architecture level power simulator, and Li et al. [48] propose a deterministic clock gating scheme. Neither work compares the efficiency of different clock gating schemes, nor explores the thermal effects of clock gating. In this chapter, by comparing the power and thermal efficiency of three different clock gating schemes, we will see it is important to take architectural factors into consideration when the clock gating decision is made.

This chapter focuses on clock gating techniques applicable to queue/ array structures in CPUs. Queue/array structures, like register files, TLBs, and every kind of decoupling queue in the processor, consume a large portion of the chip area and power budget. The chip's hotspot is typically in one of these structures. Power and thermal effects of different clock gating schemes for queue/array structures are therefore an important area of investigation.

This chapter investigates two design styles and three clock gating schemes for queue/array

structures. Two clock gating schemes apply to latch-mux design: valid-bit clock gating, in which only valid entries are clocked; and "stall" gating, in which even valid entries are not clocked when not in use. The third clock gating style applies to SRAM designs, and simply gates ports not in use. The effectiveness of valid-bit gating is determined by the queue occupancy, SRAM port gating by access rate, and latch-mux stall gating by both. The ratio of queue occupancy versus array access rate depends on architectural factors and varies from unit to unit on the chip and benchmark to benchmark.

While there are many considerations as to what design style each queue/array structure should adopt, this chapter focuses on their architectural characteristics. More specifically, this chapter investigates two architectural aspects of each structure: occupancy and access rate. If a structure has high occupancy but relatively low access rate, an SRAM-based design will be power-efficient because most of the time the structure can be clock-gated (due to its low access rate). On the other hand, if a structure usually has very few valid entries, which are accessed very frequently, then a latch-mux design makes more sense since most of the entries can be gated-off most of the time.

This chapter presents results of circuit simulations for several implementations of array structures and architectural analysis of the utilization of these structures. Despite the power and area benefits of SRAM-based array structures, there are several reasons why designers may favor latch-mux designs for relatively small array structures such as queues and buffers within a microprocessor. SRAM designs typically require a full-custom design methodology and can require additional design attention due to increased SER-susceptibility and complications with SOI process technologies. For example, array design effects with SOI technology include parasitic bipolar currents during writes and bitline leakage during read operation [1]. Latch-based design structures may also be favored as they fit in more easily with standard scan-chain based testing strategies. This work is also published in [51].

	FXQ	FPQ	FX MAPPER	FP MAPPER	FX REG	FP REG	LRQ	SRQ	SDQ
Read/Write ports number	2/2	2/2	5/2	2/1	5/2	2/1	2/2	2/2	1/1

Table 4.1: **Number of ports modeled. For clustered structures, we only report ports for a single instance [51].**

4.2 Modeling Methodology

4.2.1 Benchmarks

To keep data collection and presentation tractable, eight SPEC2000 integer benchmarks and seven SPEC2000 floating point benchmarks are used in this work. They were selected to provide a mix of programs with a range of compute-intensive vs. memory-bound behaviors.

4.2.2 Power Model

This work studies the fixed point register file (FX_REG), fixed point issue queue (FXQ), fixed point register mapping unit (FX_MAPPER), floating point register file (FP_REG), floating point issue queue (FPQ), floating point register mapping unit (FP_MAPPER), load reorder queue (LSU_LRQ), store queue (LSU_SDQ) and store reorder queue (LSU_SRQ). The number of ports modeled for these structures appears in Table 4.1. For these structures, detailed models are developed to compare the unconstrained power for SRAM and latch-mux implementations.

For the specific structures this work studied, the SRAM designs were adapted from low-power memory designs. The design utilizes minimum sized transistors and does not include sense amps because this work is primarily looking at relatively small queues and buffers. The latch-mux designs were developed specifically for this work to be as comparable as possible to the SRAM designs. The decoders and input latches were actually reused from the SRAM designs, and the latch-mux designs followed similar sizing and fanout methodology. Simulations of the latch-mux and SRAM register files were completed using Nanosim with accuracy equivalent to HSPICE. Each register file size was designed at the schematic level, for a total of eighteen designs. Designs were simulated using 130nm process technology models, at 1.2V, and 1GHz. Additionally, for the latch-mux design, the valid bits were generated externally to facilitate rapid testing. During simulation each netlist was

paired with three different vector files, corresponding to the three different measurements: read, write, and idle powers. The simulation vector files allowed Nanosim to verify the functionality of a register file while collecting power consumption data. To ensure measurement consistency, the same vector files were used to simulate SRAM and latch-mux designs of equal dimensions, based on word size and number of wordlines. Furthermore, some care was taken to ensure that different sized register files had similar input vectors.

For each design style, 9 configurations are simulated: 8, 16, and 32 bits wide for each of 8, 16, and 32 wordlines/entries. For the latch-mux designs, these simulations are repeated for scenarios with all, half, and zero entries valid. Interpolation/extrapolation are used to find the correct power for each structure of interest. These values are scaled proportionally for multi-ported structures – see Table 4.1. This work assumes 80-entry register files consist of two 40-entry banks.

4.2.3 Clock Gating Methodology

There are several styles of clock gating that we can apply. These include valid and stall gating for latch-based structures and read and write port gating for array structures.

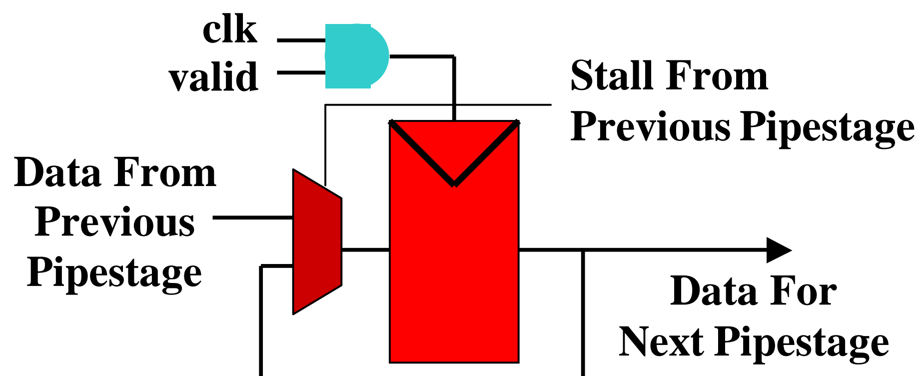


Figure 4.1: **Abstract diagrams of valid-bit gating [51].**

Figure 4.1 conceptually diagrams valid-bit based clock gating. This type of clock gating is commonly used in pipeline latches and relatively small memory structures that are designed using latch-mux schemes (e.g. issue queues, instruction buffers, etc). In this style of gating, a valid bit is associated with every bank of latches and the local clock buffer of the latch bank is gated when the

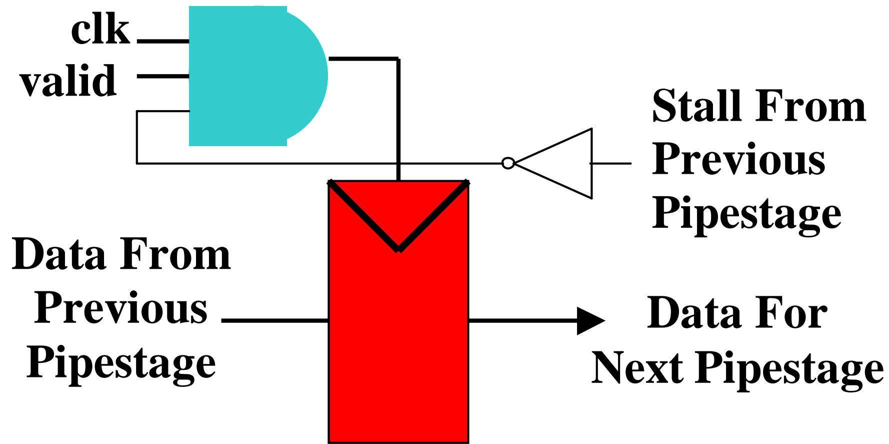


Figure 4.2: **Abstract diagrams of stall gating [51].**

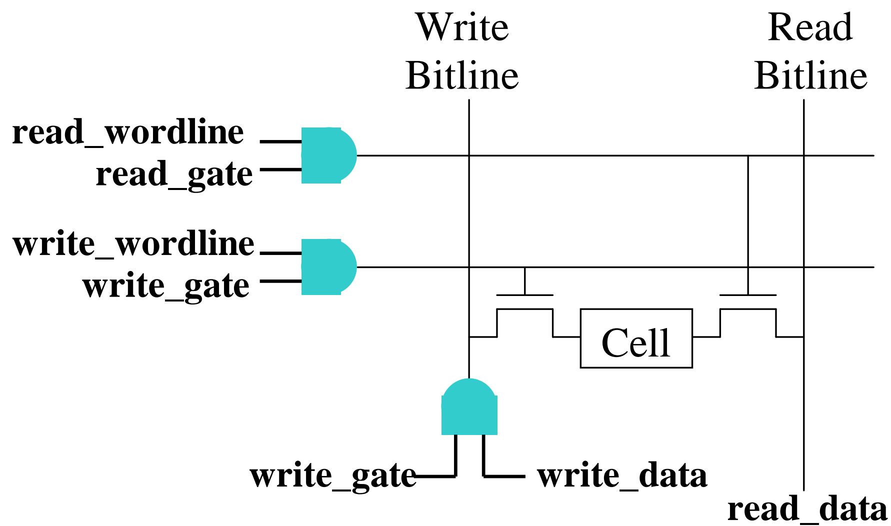


Figure 4.3: **Abstract diagrams of array gating [51].**

valid-bit is not set. Figure 4.2 diagrams stall gating, a more aggressive version of valid-bit gating, that can also clock gate a bank of latches if it is encountering a stall condition. In this case, if a bank of latches contains valid data, but the pipeline is stalled (or when a queue entry is not being accessed), the clock feeding the latch can still be gated, holding the data. While the second style of clock gating does save additional power, it requires additional timing and verification efforts; for example, the gate signal must be glitch-free. These efforts must be justified by the potential power savings quantified by architectural simulations.

Figure 4.3 conceptually diagrams the clock gating methodology that is applied to SRAM-based array structures. In this case, the array structure utilization is proportional to the number of read and write accesses to the structure. This is call as read-write port gating.

To model clock gating, it is assumed that the SRAM array and read-write circuitry can be gated, while the D-latch, precharge, and decoder circuitry cannot; and the latch-mux array can be gated but the D-latch and decoder circuitry cannot.

4.3 Results

Three clock gating styles (valid-bit gating and stall gating for latch-mux designs and read-write port gating for the SRAM design) are simulated for the units introduced in Section 4.2.2. These units can likely be implemented with either design style, but the SRAM implementation is considered more difficult to design and verify.

This section first compares the impact of the different schemes on power, then temperature. I round out the discussion by explaining the architectural behavior that favors one or the other implementation.

4.3.1 Power

Figure 4.5 compares the power dissipation of these CPU structures with different clock gating choices. These data are averaged across the integer benchmarks and the floating point benchmarks separately. (Note that even in the integer benchmarks, the floating-point mapper and register file

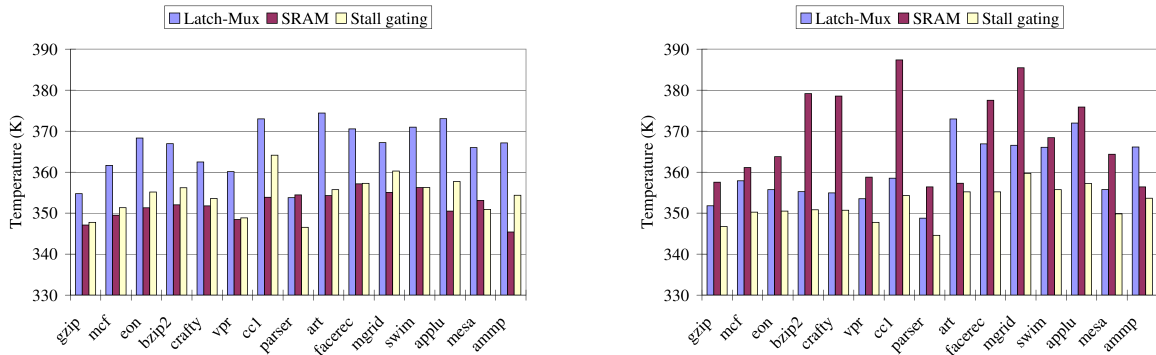


Figure 4.4: **The peak temperature of each benchmark with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]**

must hold at least 32 active registers, corresponding to the 32 FP registers in the instruction set.) Because the unconstrained power of an SRAM design is much lower than that for the corresponding latch-mux designs, the SRAM design is almost always superior, regardless of clock gating choice.

There are some important exceptions, however. The most striking exception is the fixed-point issue queue, where the latch-mux designs, even with mere valid-bit gating, are superior. The reason for this is that queues with sufficiently low occupancy favor latch-mux designs in which only active entries are clocked. As we can see from Figure 4.3.1, unlike other units, the utilization of FXQ with latch-mux design and valid-bit gating is lower than that with SRAM design. (Note that the occupancy is the same across all designs; since this work does not consider dynamic thermal management here, the different design choices do not affect execution. What matters is how the power and temperature for different design styles depend on occupancy and activity factors.)

If we compare the fixed point issue queue and the fixed point register file, entries in the register file typically must stay active much longer than in the issue queue. A fixed point instruction is put into the issue queue after renaming and is pulled out of that queue as soon as all its data dependencies are resolved. However, the entry of a physical register file can only be freed after the corresponding instruction commits. Branch mispredictions also play an important role in regularly clearing the queue and keeping average occupancy low, whereas at least 32 registers must remain active even after a misprediction flush. These factors are less true for FP programs, where mispredictions are much less frequent and FP execution latencies increase issue-queue waiting times.

Because of its low occupancy, the fixed-point issue queue favors latch-mux design for many benchmarks, despite its large unconstrained power consumption. The FXQ favors latch-mux even more with stall gating. Indeed, stall gating is always vastly superior than valid-bit gating, because stall gating can gate more entries. Even structures with high occupancies will fare well with stall gating if access rates are low.

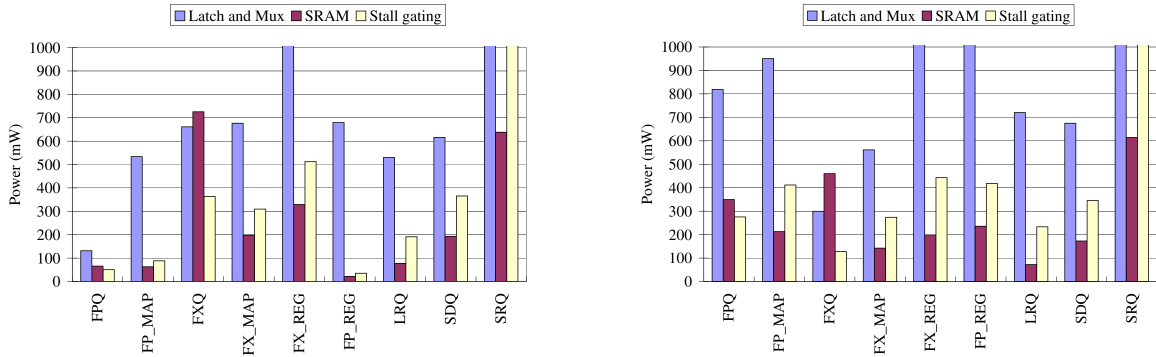


Figure 4.5: **The average unit power of integer benchmarks (left) and floating point benchmarks (right) [51]**

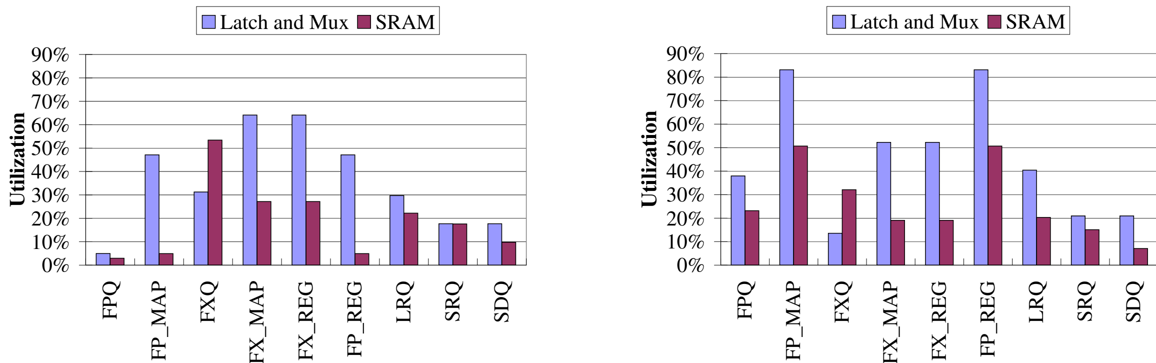


Figure 4.6: **The average unit utilization of integer benchmarks (left) and floating point benchmarks (right) [51]**

4.3.2 Temperature

As figures in the left columns of Figure 4.7 (integer workloads) and 4.8 (floating point workloads) show, if we assume that the SRAM and latch-mux designs have equal area, then the temperature follows approximately from its power. The unit temperature with SRAM design is consistently cooler than that with the latch-mux design, regardless of its clock gating styles. Even for the fixed-

point issue queue, although the power consumption of this structure with SRAM design is higher than with the latch-mux design, its temperature is lower due to thermal coupling with neighboring units, which all have consistently higher power consumption and higher temperatures with the latch-mux design. Considering the thermal profile of each possible combination is beyond the scope of this work but necessary to fully consider the interaction of design style and thermal coupling.

Of course, the SRAM design is likely smaller than the latch-mux design. This increases its power density. From the circuit design, it is estimated that the same frequency SRAM design is roughly 3.3 times smaller than the corresponding latch-mux design. If this area effect is included, we will have the units temperature figures in the right column of Figure 4.7 and 4.8. As we can see from these figures, the increased power density of the SRAM design versus the lower power density of the latch-mux design increase the temperature of the units with the SRAM design and decrease the temperature of the units with the latch-mux design. Now for the latch-mux design with stall gating, temperature is consistently lower than for the SRAM design. Even for the latch-mux design with valid bit gating, the FXQ, FX_MAP, and FX_REG have lower temperatures than the SRAM design. The temperature of the SRAM design can be reduced by enlarging its area, however, this will lead to extra latency. It is the future work to quantify this temperature/performance tradeoff with area scaling for the SRAM design.

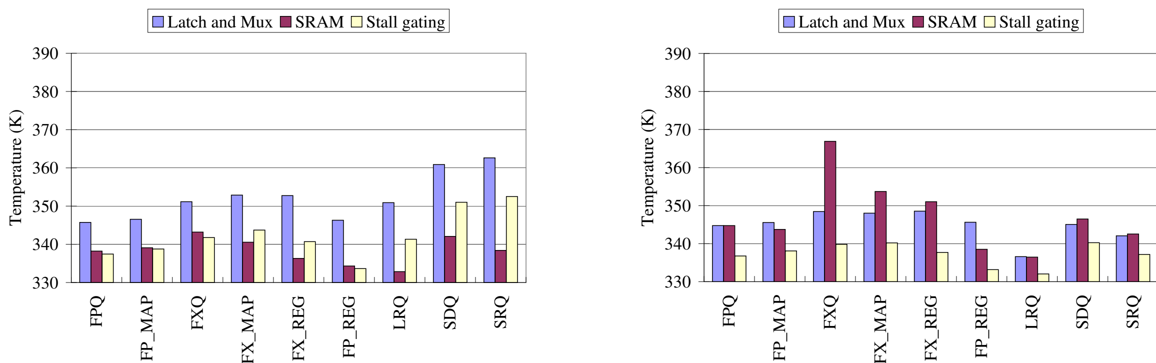


Figure 4.7: **The temperature of the units for integer benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]**

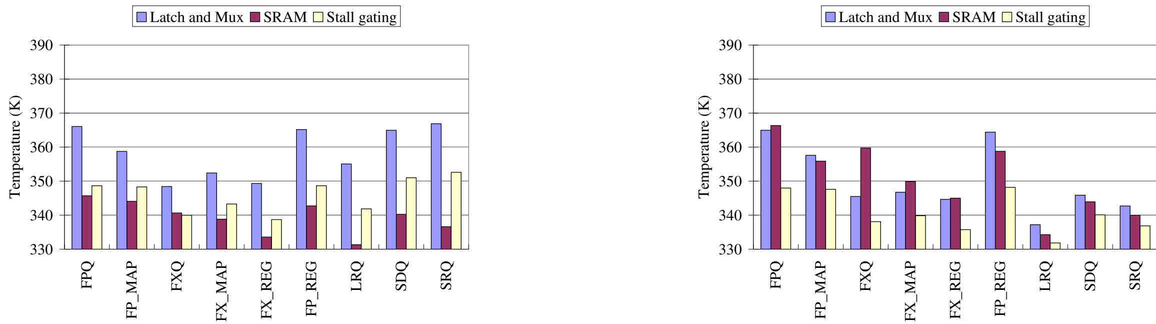


Figure 4.8: **The temperature of the units for floating point benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]**

4.3.3 Per-Benchmark Differences

The relative power and thermal efficiency of different clock gating styles not only changes from unit to unit, but also changes from benchmark to benchmark.

Figure 4.9 illustrates this trend for the fixed-point issue queue. As we can see from this figure, we can classify the four benchmarks into four categories: mcf has high occupancy, low access rate; crafty has low occupancy, high access rate; gcc has high occupancy, high access rate; and art has low occupancy, low access rate. Corresponding to these different occupancy-access rate ratios, for the latch-mux design with valid bit gating, mcf and gcc have relatively high temperatures while crafty and art have relatively low temperatures; while for the SRAM design, crafty and cc1 have relatively high temperatures and mcf and art have relatively low temperatures.

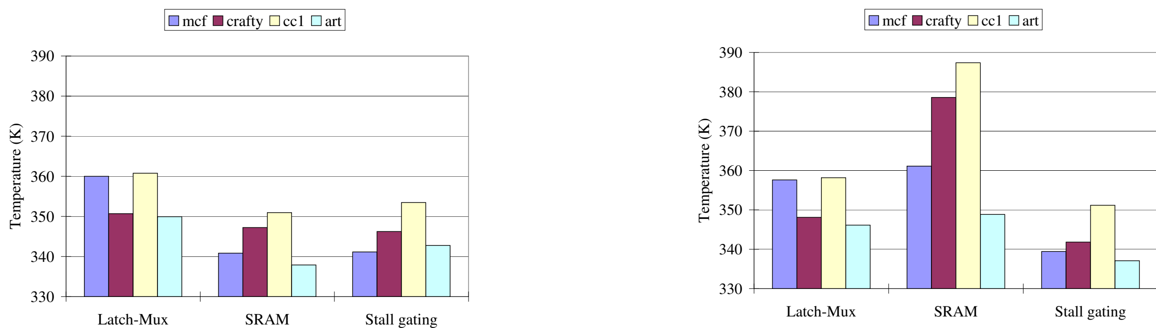


Figure 4.9: **The temperature of FXQ for four benchmarks with the ratio of the area of the Latch-Mux design versus the SRAM design at 1 (left) and at 3.3 (right) [51]**

4.4 Future Work and Conclusions

This chapter investigates energy and thermal effects of different design styles and their associated clock gating choices for queue and array structures in a high-performance, superscalar, out-of-order CPU. SRAM and latch-mux structures are simulated to determine their power dissipation as well as their scaling properties. Then these data are used in architectural cycle-accurate performance/power/thermal simulations.

The SRAM and latch-mux designs only represent one possible set of designs. While the specific implementations, areas, and resultant hotspots may vary with different designs, this chapter illustrates intrinsic differences between SRAM and latch-mux designs. Specifically, this chapter finds that even though SRAM designs have a huge advantage according to their unconstrained power, results can be different when architecture-level effects are modeled. Even latch-mux designs with valid-bit gating, the worst of all three designs, outperforms SRAM for a queue with low occupancy but high access rate, namely the integer issue queue. Furthermore, even though SRAM designs do yield the lowest power dissipation for most structures, their smaller area leads to higher power density. Assuming a 3X area ratio, this causes latch-mux designs with stall gating to consistently give better thermal performance for most structures and most benchmarks.

These results show that circuit-level simulations are insufficient for making design-style and clock-gating choices. The behavior of these structures also depends on architecture-level and thermal behavior. Especially in an era of thermally limited design, latch-mux designs with stall gating are an attractive choice, despite their apparent disadvantage when viewed purely from the perspective of raw switching power. SRAMs also have other implementation and testing drawbacks.

Finally, this work shows the importance of considering design style and clock gating for thermal simulation, as they substantially change operating temperatures and the distribution of hot spots.

The current results apply to relatively small queue/buffer structures. Scaling to larger structures, exploring designs of different densities (to trade off performance for reduced power density), and a more detailed exploration of how thermal coupling affects these design decisions are all interesting areas for future work.

Chapter 5

Performance, Energy and Temperature Considerations for CMP and SMT architectures

5.1 Introduction

Simultaneous multithreading (SMT) [76] is a recent microarchitectural paradigm that has found industrial application [35, 55]. SMT allows instructions from multiple threads to be simultaneously fetched and executed in the same pipeline, thus amortizing the cost of many microarchitectural structures across more instructions per cycle. The promise of SMT is area-efficient throughput enhancement. But even though SMT has been shown energy efficient for most workloads [52, 66], the significant boost in instructions per cycle (IPC) means increased power dissipation and possibly increased power density. Since the area increase reported for SMT execution is relatively small (10-20%), thermal behavior and cooling costs are major concerns.

Chip multiprocessing (CMP) [24] is another relatively new microarchitectural paradigm that has found industrial application [35, 39]. CMP instantiates multiple processor "cores" on a single die. Typically the cores each have private branch predictors and first-level caches and share a second-level, on-chip cache. For multi-threaded or multi-programmed workloads, CMP architectures amortize the cost of a die across two or more processors and allow data sharing within a common L2 cache. Like SMT, the promise of CMP is a boost in throughput. The replication of cores means that the area and power overhead to support extra threads is much greater with CMP

than SMT. For a given die size, a single-core SMT chip will therefore support a larger L2 size than a multi-core chip. Yet the lack of execution contention between threads typically yields a much greater throughput for CMP than SMT [10, 24, 64]. A side effect is that each additional core on a chip dramatically increases its power dissipation, so thermal behavior and cooling costs are also major concerns for CMP.

Because both paradigms target increased throughput for multi-threaded and multi-programmed workloads, it is natural to compare them. This chapter provides a thorough analysis of the performance benefits, energy efficiency, and thermal behavior of SMT and CMP in the context of a POWER4-like microarchitecture. This research assumes POWER4-like cores with similar complexity for both SMT and CMP except for necessary SMT related hardware enhancements. Although reducing the CMP core complexity may improve the energy and thermal efficiency for CMP, it is cost effective to design a CMP processor by reusing an existing core. The POWER5 dual SMT core processor is an example of this design philosophy.

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

It is also found that the thermal profiles are quite different between CMP and SMT architectures. With the CMP architecture, the heating is primarily due to the global impact of higher energy output. For the SMT architecture, the heating is very localized, in part because of the higher utilization of certain key structures such as the register file. These different heating patterns are critical when we need to consider *dynamic thermal management* (DTM) strategies that seek to use runtime control

to reduce hotspots. In general, this work finds that DTM strategies which target local structures are superior for SMT architectures and that global DTM strategies work better with CMP architectures.

The rest of the chapter is organized as follows. Section 5.2 discusses the related work in comparing SMT and CMP processors from an energy-efficiency standpoint. Section 5.3 discusses the details of the performance, power, and temperature methodology that is utilized in this work, including the choice of L2 sizes to study. Section 5.4 discusses the baseline results for SMT and CMP architectures without DTM. Section 5.5 explores the more realistic case when microprocessors are DTM constrained and explores which strategies are best for CMP and SMT under performance and energy-constrained designs. Section 5.6 concludes this chapter and discusses avenues for future research. This work is also published in [54].

5.2 Related Work

There has been a burst of work in recent years to understand the energy efficiency of SMT processors. We [52] study the area overhead and energy efficiency of SMT in the context of a POWER4-like microarchitecture, and Seng et al. [66] study energy efficiency and several power-aware optimizations for a multithreaded Alpha processor. Sasanka et al. consider the energy-efficiency of SMT and CMP for multimedia workloads [64], and Kaxiras et al. [37] do the same for mobile phone workloads on a digital signal processor. Like this work does, these other studies find that SMT boosts performance substantially (by about 10–40% for SPEC workloads), and that the increase in throughput more than makes up for the higher rate of power dissipation, with a substantial net gain in energy efficiency.

For multithreaded and multiprogrammed workloads, CMP offers clear performance benefits. If contention for the second-level cache is not a problem, speedups are close to linear in the number of cores. Although energy efficiency of CMP organizations have been considered for specific embedded-system workloads, the energy efficiency of CMP for high-performance cores and workloads has not been well explored. Sasanka et al. consider the energy-efficiency of SMT and CMP for multimedia workloads [64], and Kumar et al. [41] consider energy efficiency for a heteroge-

neous CMP core, but only for single-threaded workloads. Like this work does, these other studies both find substantial energy benefits.

Other researchers have compared SMT and CMP. Sasanka et al., Kaxiras et al., Kumar et al. [44], Burns et al. [10], and Hammond et al. [24] all find that CMP offers a substantial performance advantage when there are enough independent threads to keep all cores occupied. This is generally true even when the CMP cores are simpler than the SMT core—assuming enough thread-level parallelism to take advantage of the CMP capability.

Several authors [10,44,64] also consider hybrids of SMT and CMP (e.g., two CMP cores, each supporting 2-way SMT), but with conflicting conclusions. They generally find a hybrid organization with N thread contexts inferior to CMP with N full cores, but to differing degrees. It is unclear to what extent these conclusions hold true specifically for memory-bound workloads. Since CMP seems superior to a hybrid organization, this work focuses only on purely 2-way SMT (one core) and 2-way CMP systems (one thread per core) in order to focus on the intrinsic advantages of each approach. While a study of the combined energy and thermal efficiency of hybrid CMP/SMT systems is interesting, it is beyond the scope of this chapter: the incredibly complex design space described by [10,44,64] means that analyzing this configuration can easily occupy an entire chapter by itself. In any case, understanding the combined energy and thermal efficiency of plain SMT and CMP systems is a prerequisite, and except for the work by Sasanka et al. and Kaxiras et al. for specialized workloads, there is no other work comparing the energy efficiency of SMT and CMP. Sasanka et al. find CMP to be much more energy efficient than SMT, while Kaxiras et al. find the reverse. The reason is that the Sasanka work uses separate programs which scale well with an increasing number of processors and can keep all processors occupied. In contrast, with the mobile phone workload of Kaxiras et al., not all threads are active all the time, and idle cores waste some energy. Instead, their SMT processor is based on a VLIW architecture and is wide enough to easily accommodate multiple threads when needed.

I am only aware of two other papers exploring thermal behavior of SMT and/or CMP. Heo et al. [26] look at a variety of ways to use redundant resources, including multiple cores, for migrating computation of a single thread to control hot spots, but find the overhead of core swapping is high.

Donald and Martonosi [17] compare SMT and CMP and find that SMT produces more thermal stress than CMP. But, like many other studies comparing SMT and CMP, their analysis assumes that the cores of the CMP system are simpler and have lower bandwidth than the single-threaded and SMT processors, while this work follows the pattern of the IBM POWER4/POWER5 series and assumes that all three organizations offer the same issue bandwidth per core. Donald and Martonosi also consider a novel mechanism to cope with hotspots, by adding "white space" into these structures in a checkerboard fashion to increase their size and hopefully spread out the heat, but found that even a very fine-grained partitioning did not achieve the desired heat spreading. This work adopts a similar idea for the register file, the key hotspot, but rather than increase its size, this work throttles its occupancy. Simulations using an improved version of HotSpot in [31] suggest that sufficiently small structures will spread heat effectively.

5.3 Modeling Methodology

According to [14], the POWER5 offers 24 sensors on chip. Accordingly, we can assume it is reasonable to provide at least one temperature sensor for each microarchitecture block in the floorplan, and that these sensors can be placed reasonably close to each block's hotspot, or that data fusion among multiple sensors can achieve the same effect. We can also assume that averaging and data fusion allow dynamic noise to be ignored , and that offset errors can be removed by calibration [3]. The temperature is sampled every 100k cycles and set DTM experiments' thermal emergency threshold at 83°C. This threshold is carefully chosen so for single thread single core architecture it will normally lead to less than 5% performance loss due to DTM control. At the beginning of the simulation, the steady state temperature is set for each unit as the initial temperature so the whole simulation's thermal output will be meaningful. For DTM experiments, the initial temperature is set as the smaller value of the steady state temperature without DTM and the thermal emergency threshold which is 83°C in all DTM experiments.

	gzip	mcf	eon	bzip2	crafty	vpr	cc1	parser
IPC	L	L	H	H	H	H	H	L
temperature	L	H	L	H	H	L	H	L
L2 miss ratio	L	H	L	L	L	L	L	L

Table 5.1: **Categorization of integer benchmarks [54]**

5.3.1 Benchmark Pairs

15 SPEC2000 benchmarks are used as single thread benchmarks. They are compiled by the *xlc* compiler with the -O3 option. First the Simpoint toolset [67] is used to get representative simulation points for 500-million-instruction simulation windows for each benchmark, then the trace generation tool generates the final static traces by skipping the number of instructions indicated by Simpoint and then simulating and capturing the following 500 million instructions.

Pairs of single-thread benchmarks are used to form dual-thread SMT and CMP benchmarks. There are many possibilities for forming the pairs from these 15 benchmarks. The following methodology are utilized to form pairs. First, each single thread benchmark combines with itself to form a pair. Also several SMT and CMP benchmarks are formed by combining different single thread benchmarks. Here the single thread benchmarks are categorized into eight major categories: high IPC (> 0.9) or low IPC (< 0.9), high temperature (peak temperature $> 82°C$) or low temperature (peak temperature $< 82°C$), floating benchmark or integer benchmark as shown in Table 5.1 and 5.2.

Then eighteen pairs of dual-thread benchmarks are formed by selecting various combinations of benchmarks with these characteristics. Note that the choice of memory-bound benchmarks was limited. This is a serious drawback to using SPEC for studies like this. The architecture community needs more benchmarks with a wider range of behaviors.

The rest of this chapter discusses workloads in terms of those with high L2 cache miss ratio vs. those with low L2 cache miss ratio. When one benchmark in a pair has a high L2 cache miss ratio, that pair is categorized as a high L2 cache miss pair.

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

Table 5.2: **Categorization of floating point benchmarks [54]**

5.3.2 Chip Die Area and L2 Cache Size Selection

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

In the 2004-2005 timeframe, mainstream desktop and server microprocessors include aggressive, out-of-order processor cores coupled with 512KB to 2MB of on-chip L2 cache. The experiments indicate that for very large L2 cache sizes and typical desktop and workstation applications (SPEC2000), most benchmarks will fit in the cache for both the SMT and CMP machines. But for a fixed number of cores, Figure 5.1 shows that as die size is reduced, SMT eventually performs better than CMP for memory-bound benchmarks. This is because a core occupies about 1 MB's worth of space, so SMT's L2 sizes are 1 MB larger than CMP's. Given constraints on chip area, it is likely that there will always be certain memory-bound workloads that will perform better with SMT than with CMP. Recognizing this tradeoff, the L2 cache is set at 1MB for CMP and at 2MB for SMT for the baseline study and discuss where appropriate how these choices impact conclusions.

5.4 Baseline Results

This section discusses the performance, energy, and temperature implications of SMT and CMP designs *without* dynamic thermal management. The next section considers thermally limited designs.

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

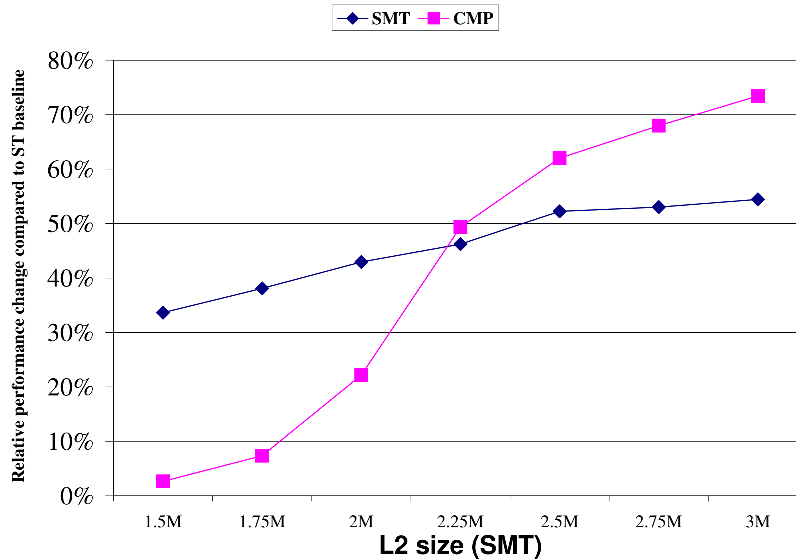


Figure 5.1: **Performance of SMT and CMP for memory-bound benchmarks (the categorization is done with 2MB L2 cache size for ST) with different L2 cache size [54]**

at 210 mm^2 including the on-chip level two cache. This means CMP will have the smallest L2 cache, since its core area is the largest among the three. In this work, the L2 cache sizes for ST, SMT, and CMP are 2MB, 2MB, and 1MB respectively. Because the SMT core core is only 12% larger than the ST core, both use 2MB L2 cache.

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

5.4.1 SMT and CMP Performance and Energy

Figure 5.2 breaks down the performance benefits and energy efficiency of SMT and CMP for the POWER4-like microarchitecture. The results in this figure are divided into two classes of benchmarks – those with relatively low L2 miss rates (left) and those with high L2 cache miss rates (right). This figure shows that CMP dramatically outperforms SMT for workloads with low to modest L2 miss rates, with CMP boosting throughput by 87% compared to only 26% for SMT. But the CMP chip has only half the L2 cache as SMT, and for workloads with high L2 miss rate, CMP only affords a throughput benefit of 22% while SMT achieves a 42% improvement.

The power and energy overheads demonstrated in Figure 5.2 are also enlightening. The power

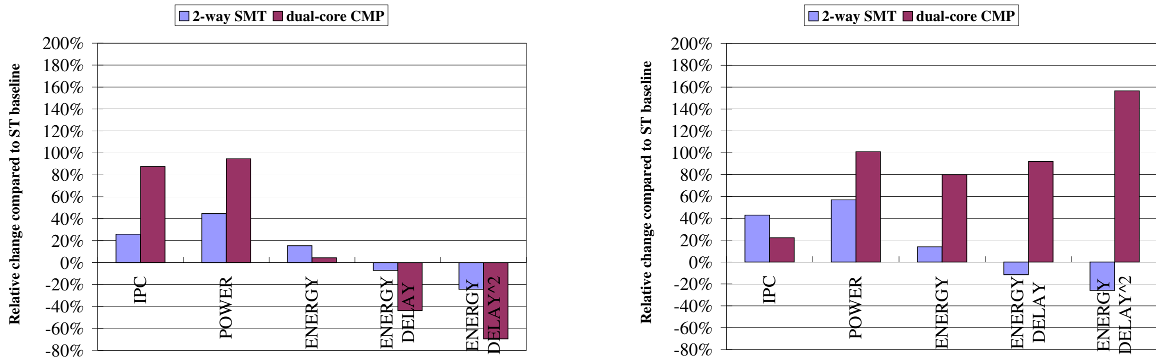


Figure 5.2: **Performance and Energy efficiency of SMT and CMP compared to ST, for low L2 cache miss workloads (left) and high L2 cache miss workloads (right) [54].**

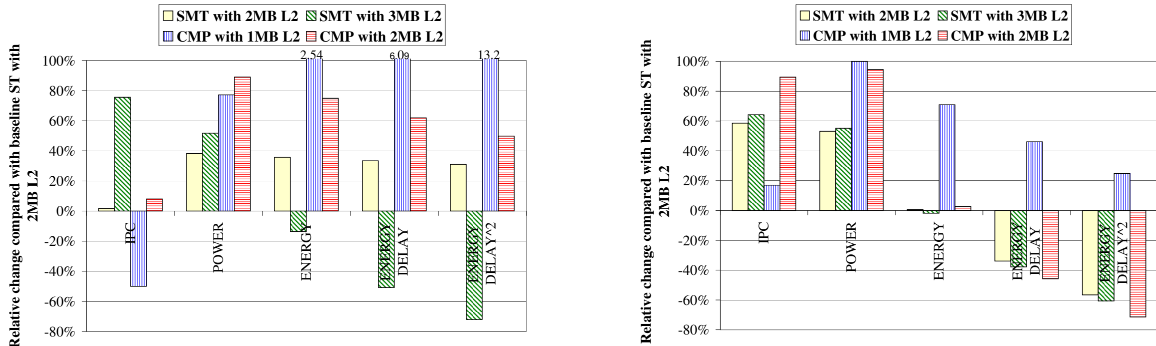


Figure 5.3: **Performance and Energy efficiency of SMT and CMP compared to ST as L2 size changes. On the left are results for a benchmark (mcf+mcf) which is memory bound for all L2 configurations shown. On the right are results for a benchmark (mcf+vpr) which ceases to be memory-bound once L2 size changes from 1MB to 2MB for CMP [54].**

overhead of SMT is 45–57%. The main reasons for the SMT power growth are the increased resources that SMT requires (e.g. replicated architected registers), the increased resources that are needed to reduce new bottlenecks (e.g. additional physical registers), and the increased utilization due to additional simultaneous instruction throughput [52]. The power increase due to CMP is even more substantial: 95% for low-L2-miss-rate workloads and 101% for the high-miss-rate workloads. In this case the additional power is due to the addition of an entire second processor. The only reason the power does not double is that L2 conflicts between the two cores lead to stalls where clock gating is engaged, and this explains the lower power overhead of the L2-bound workloads.

Combining these two effects with the energy-delay-squared metric (ED^2) [80], we see that CMP is by far the most energy-efficient organization for benchmarks with reasonable L2 miss rates, while SMT is by far the most energy-efficient for those with high miss rates. Indeed, for L2-bound workloads, from the standpoint of ED^2, a single-threaded chip would be preferable to CMP, even though the single-threaded chip cannot run threads in parallel. Of course, this is at least in part due to the reduced L2 on the CMP chip.

When we increase L2 cache size, some benchmarks that had previously been memory bound now fit better in the L2 cache, and thus need to be categorized as low L2 miss rate benchmarks. Figure 5.3 illustrates the consequences. The graph on the right shows how mcf+vpr ceases to be memory bound when we increase the L2 cache sizes by 1 MB (SMT from 2MB to 3MB and CMP from 1MB to 2MB). With smaller L2 cache size and high cache miss ratio, the program is memory-bound and SMT is better in terms of performance and energy efficiency. With larger L2 size and low cache miss ratio, the program is no longer memory bound and CMP is better. Of course, for any L2 size, some applications' working set will not fit, and these benchmarks will remain memory bound. The left-hand graph in Figure 5.3 illustrates that SMT is superior for memory-bound benchmarks.

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

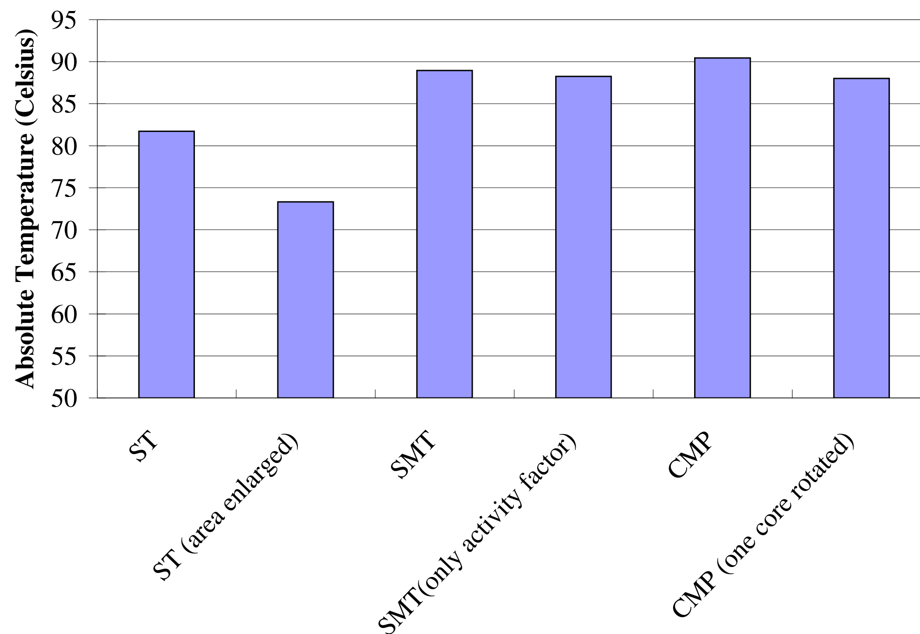


Figure 5.4: **Temperature of SMT and CMP vs. ST [54]**

5.4.2 SMT and CMP Temperature

Figure 5.4 compares the maximum measured temperature for several different microprocessor configurations. We see that the single-threaded core has a maximum temperature of nearly 82°C. When we consider the SMT processor, the temperature increases around 7 degrees and for the CMP processor the increase is around 8.5 degrees.

With such a small difference in temperature, it is difficult to conclude that either SMT or CMP is superior from a temperature standpoint. In fact, if we rotate one of the CMP cores by 180 degrees, so the relatively cool IFU of core 1 is adjacent to the hot FXU of core 0, the maximum CMP processor temperature will drop by around 2 degrees, which makes it slightly cooler than the SMT processor.

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

If we take the SMT core and only scaled the power dissipation with increased utilization (omit-

ting the increased power dissipation due to increased resources and leaving the area constant). From Figure 5.4 we can see that the SMT temperature will rise to nearly the same level as when all three factors are included. This makes sense when we consider that the *unconstrained power density* of most of the scaled structures in the SMT processor (e.g. register files and queues) will likely be relatively constant because the power and area will both increase with the SMT processor, and in this case the utilization increase becomes the key for SMT hotspots. From this we can conclude that for the SMT processor, the temperature hotspots are largely due to the higher utilization factor of certain structures like the integer register file.

The reasoning behind the increase in temperature for the CMP machine is quite different. For the CMP machine, the utilization of each individual core is nearly the same as for the single-thread architecture. However, on the same die area we have now integrated two cores and the total power of the chip nearly doubles (as we saw in Figure 5.2) and hence the total amount of heat being generated nearly doubles. Because of the large chip-level energy consumption, the CMP processor heats up the TIM, heat spreader, and heat sink, thus raising the temperature of the overall chip. Thus the increased temperature of the CMP processor is due to a global heating effect, quite the opposite of the SMT processor's localized utilization increase. This fundamental difference in thermal heating will lead to substantial differences in thermal trends as we consider future technologies and advanced dynamic thermal management techniques.

5.4.3 Impact of Technology Trends

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

Figure 5.5 shows the impact of technology scaling on the temperature of SMT and CMP processors. This figure shows the difference in absolute temperature between the CMP and SMT core for three generations of leakage (roughly corresponding to 130nm, 90nm, and 70nm technologies).

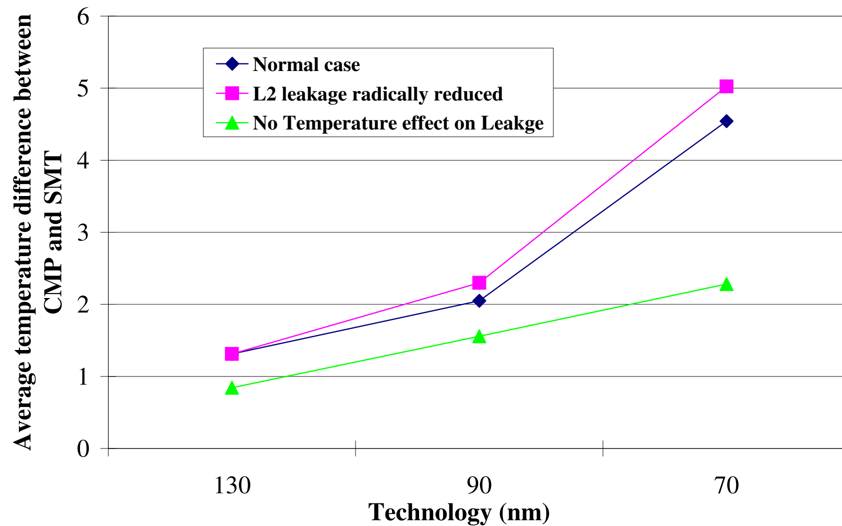


Figure 5.5: **Temperature Difference between CMP and SMT for different technologies [54]**

As we project towards future technologies, there are several important trends to note. The most important trend is that the temperature difference between the CMP machine (hotter) and SMT machine (cooler) increases from 1.5 degrees with the baseline leakage model to nearly 5 degrees with the most leaky technology. The first reason for this trend is that the increased utilization of the SMT core becomes muted by higher leakage. The second reason is that the SMT machine's larger L2 cache tends to be much cooler than the second CMP core. This, coupled with the exponential temperature dependence of subthreshold leakage on temperature, causes the CMP processor's power to increase more than the SMT processor. This aggravates the CMP processor's global heat up effect. From Figure 5.5, we can see that if we remove the temperature dependence of leakage in the model, the temperature difference between the CMP and SMT machine grows much less quickly. Figure 5.5 also shows how the trend is amplified when we consider the case where aggressive leakage control is applied to the L2 cache (perhaps through high-Vt transistors). In this case, the SMT processor is favored because a larger piece of the chip is eligible for this optimization.

5.5 Aggressive DTM constrained designs

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

It is important to note that peak temperature is not indicative of cooling costs. A benchmark with short periods of very high temperature, separated by long periods of cooler operation, may incur low performance overhead from DTM, while a benchmark with more moderate but sustained thermal stress may engage DTM often or continuously. To illustrate this point, Figure 5.6 plots DTM performance loss against maximum temperature. The scattered nature of the points and poor correlation coefficients show that maximum temperature is a poor predictor of DTM overhead.

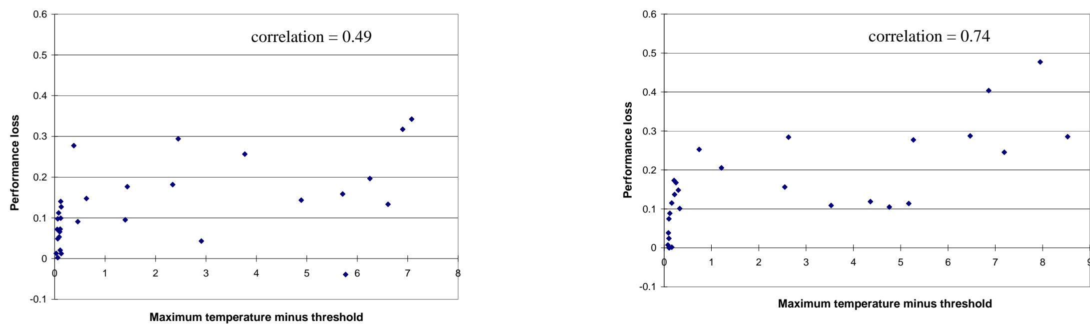


Figure 5.6: **Performance loss from DTM vs. peak temperature. Peak temperature here is plotted as the number of degrees by which the maximum temperature exceeds the trigger threshold [54]**

To make an equal comparison of DTM performance among single-threaded, SMT, and CMP chips, the same thermal package is used for all three configurations (see Section 5.3).

5.5.1 DTM Techniques

Four DTM strategies are implemented in this work:

- Dynamic voltage scaling (DVS): DVS cuts voltage and frequency in response to thermal violations and restores the high voltage and frequency when the temperature drops below the trigger threshold. The low voltage is always the same, regardless of the severity of thermal stress; this was shown in [69] to be just as effective as using multiple V/F pairs and a controller. For these workloads, a voltage of 0.87 (79% of nominal) and frequency of 1.03GHz (77% of nominal) are always sufficient to eliminate thermal violations. Because there is not yet a consensus on the overhead associated with switching voltage and frequency, this work tests both 10 and 20 μs stall times for each change in the DVS setting.

- Fetch-throttling: Fetch-throttling limits how often the fetch stage is allowed to proceed, which reduces activity factors throughout the pipeline. The duty cycle is set by a feedback controller.

- Rename-throttling: Rename throttling limits the number of instructions renamed each cycle. Depending on which register file is hotter with the outcome of the previous sampling period, either floating-point register renaming or integer register renaming will be throttled. This reduces the rate at which a thread can allocate new registers in whichever register file has overheated, and is thus more localized in effect than fetch throttling. But if the throttling is severe enough, this has the side effect of slowing down the thread that is causing the hot spot. This can degenerate to fetch throttling, but when it is the FP register file being throttled, the slowdown can be valuable for mixed FP-integer workloads by helping to regulate resource use between the two threads.

- Register-file occupancy-throttling: The register file is usually the hottest spot of the whole chip, and its power is proportional to the occupancy. One way to reduce the power of the register file is to limit the number of register entries to a fraction of the full size. To distribute the power density, we can interleave the on and off registers, so that the heat can be more evenly spreaded across the whole register file. It is important to note that the modeling of this technique here is idealistic, assuming that the reduction in power density across the register file is proportional to the number of registers that have been turned off. This assumes an ideal interleaving and ideal heat spreading and neglects power dissipation in the wiring, which

will not be affected with occupancy throttling. This technique is included to demonstrate the potential of value of directly reducing power density in the structure that is overheating, rather than reducing activity in the whole chip.

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

DVS's cubic advantage is appealing, but as operating voltages continue to scale down, it becomes more difficult to implement a low voltage that adequately cuts temperature while providing correct behavior and reasonable frequency. Another concern with DVS is the need to validate products for two voltages rather than one. Finally, the assumption that both frequency and voltage can change in 10–20 μs may be optimistic. If voltage and frequency must change gradually to avoid circuit noise, the latency to achieve adequate temperature reduction may be prohibitively long.

Register-occupancy throttling is limited to register files based on a latch-and-mux design. Power dissipation in SRAM-based designs is likely to be much more heavily dominated by the decoders, sense amplifiers. Furthermore, this technique may be idealistic, because it assumes that reducing register file occupancy uniformly reduces power density, when in fact those registers that remain active will retain the same power dissipation. But this does not mean that the temperature of active registers remains unchanged, because neighboring areas of lower power density can help active

[1] This is only an approximate relationship; experiments of this work derive the actual V-f relationship from ITRS data [68].

registers to spread their heat. Whether a register is small enough to spread enough heat laterally is an open question and requires further analysis. However, results in [31] using HotSpot 2.0 suggest that, below about 0.2–0.25 mm and for a 0.5mm die with a typical high-performance package, the ratio of vertical to lateral thermal resistance is so high that heat spreads out very quickly, without raising the localized temperature. This result differs from the findings of [17], who used HotSpot 1.0 to find that much smaller sizes are needed to spread heat. But HotSpot 1.0 omits the TIM's very high thermal resistance and performs less detailed thermal modeling of heat flow in the package. Clearly the granularity at which spreading dominates, and alternative layouts and organizations which can reduce hotspots, is an important area requiring further research. But almost all prior DTM research has focused on global techniques like fetch gating, voltage-based techniques, or completely idling the hot unit, all of which suffer from significant overheads. What is needed are techniques that can reduce power density *in situ*, without introducing stalls that propagate all the way up the pipeline. Register-occupancy throttling illustrates that such an approach offers major potential benefits, and that further research in this direction is required.

5.5.2 DTM Results: Performance

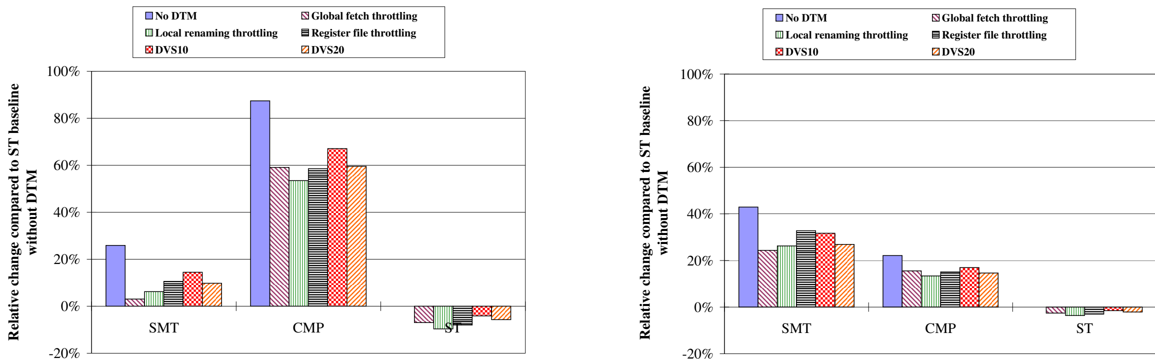


Figure 5.7: **Performance of SMT and CMP vs. ST with different DTM policies, all with threshold temperature of** 83°C**. Workloads with low L2 cache miss rate are shown on the left. Workloads with high L2 cache miss rate are shown on the right [54].**

For many traditional computing design scenarios, performance is the most critical parameter, and designers primarily care about power dissipation and thermal considerations because of thermal limits. In these cases, designers would like to optimize performance under thermal constraints.

These include systems such as traditional PC desktops and certain high-performance server environments where energy utility costs are not critical.

To evaluate architectures viable for these situations, Figure 5.7 shows performance of SMT and CMP architectures with different DTM schemes. As we observed in the previous section, the results are again dependent on whether the workloads have high L2 miss ratio. For workloads with low or moderate miss ratios, CMP always gives the best performance, regardless of which DTM technique is used. On the other hand, for workloads that are mostly memory bound, SMT always gives better performance than CMP or ST.

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

Looking at the low L2 miss workloads (Figure 5.7, left) and the high L2 miss workloads (Figure 5.7, right), we find that SMT and CMP diverge with regards to the optimal throttling scheme. For CMP, fetch-throttling and register-occupancy throttling work equally well, and both outperform local rename-throttling. For SMT, register throttling is the best performing throttling scheme, followed by rename-throttling and global fetch-throttling. In fact, for SMT running high L2 miss workloads, the local register occupancy throttling performs better than all of the other DTM techniques including DVS.

The relative effectiveness of the DTM techniques illustrates the different heating mechanisms of CMP and SMT, with heating in the CMP chip a more global phenomenon, and heating in the SMT chip localized to key hotspot structures. For example, by directly resizing the occupancy of the register file, register-throttling is very effective at reducing the localized power density of the

register file, and bringing down the temperature of the register file. In other words, the match-up between the mechanism of register-throttling and the inherent heat-up mechanism makes register-throttling the most effective DTM scheme for SMT. On the other hand, CMP mainly suffers from the global heat up effects due to the increased power consumption of the two cores. Thus global DTM schemes that quickly reduce total power of the whole chip perform best for CMP. This conclusion remains unchanged when increasing the L2 cache size to 2MB for CMP.

5.5.3 DTM Results: Energy

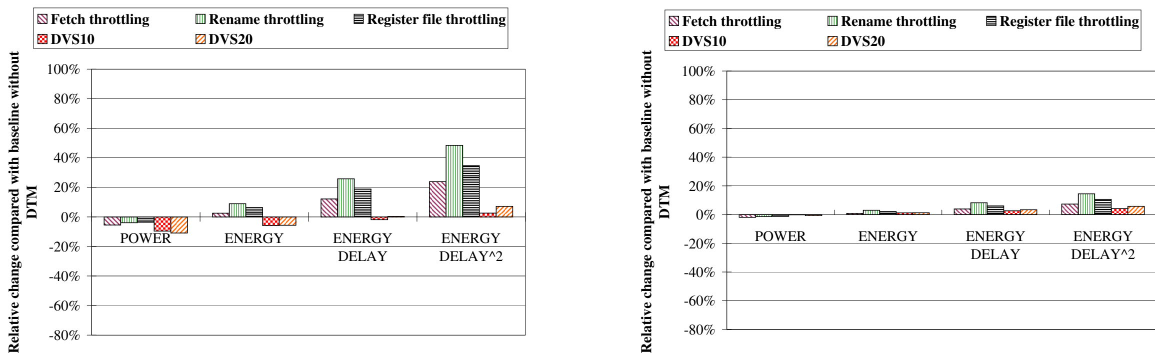


Figure 5.8: **Energy-efficiency metrics of ST with DTM, compared to ST baseline without DTM, for low-L2-miss-rate workloads (left) and high-L2-miss-rate workloads (right) [54].**

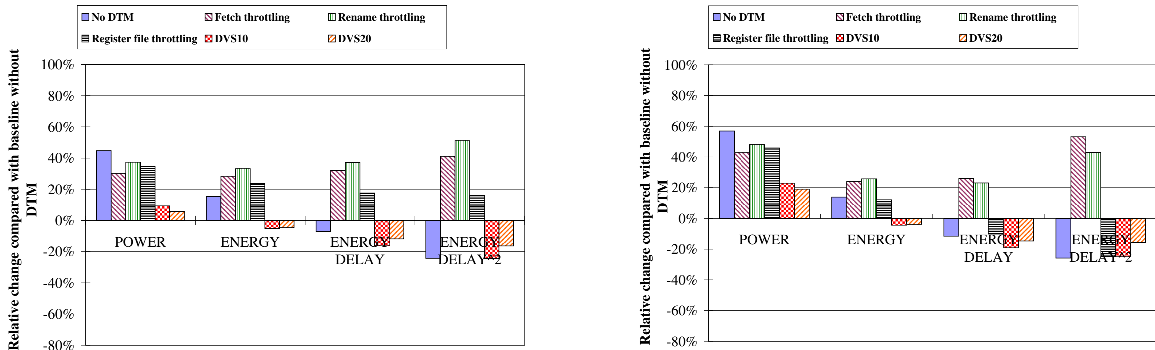


Figure 5.9: **Energy-efficiency metrics of SMT with DTM, compared to ST baseline without DTM, for low-L2-miss-rate benchmarks (left) and high-L2-miss-rate benchmarks (right) [54].**

In many emerging high-performance computing environments, designers must optimize for raw performance under thermal packaging constraints, but energy consumption is also a critical design criteria for battery life or for energy utility costs. Examples of these systems are high-performance

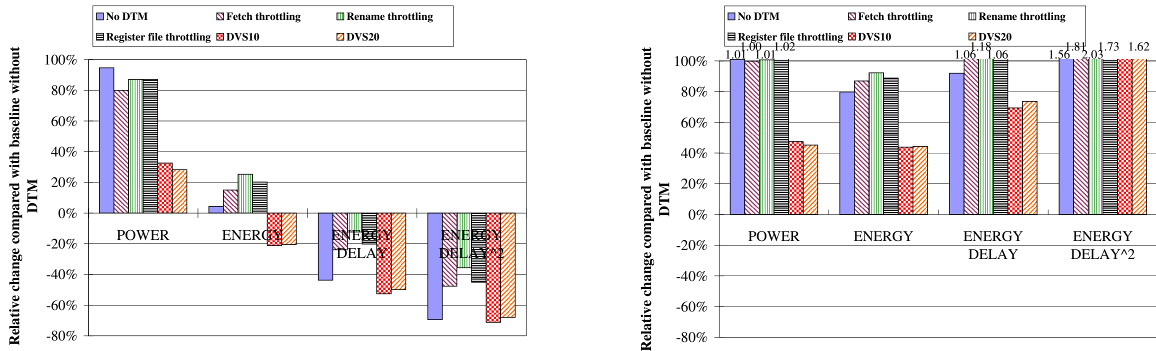


Figure 5.10: **Energy-efficiency metrics of CMP with DTM, compared to ST baseline without DTM, for low-L2-miss-rate benchmarks (left) and high-L2-miss-rate benchmarks (right) [54].**

mobile laptops and servers designed for throughput oriented data centers like the Google cluster architecture [4].

In this scenario, designers often care about joint power-performance system metrics after DTM techniques have been applied. Figure 5.8 through Figure 5.10 shows the power and power-performance metrics (energy, energy-delay, and energy-delay^2) for the ST, SMT, and CMP architectures after applying the DTM techniques. All of the results in these figures are compared against the baseline ST machine without DTM. From these figure, we see that the dominant trend is that global DTM techniques, in particular DVS, tend to have superior energy-efficiency compared to the local techniques for most configurations. This is true because the global nature of the DTM mechanism means that a larger portion of the chip will be cooled, resulting in a larger savings. This is especially obvious for the DVS mechanism, because DVS's cubic power savings is significantly higher than the power savings that the throttling techniques provide. The two local thermal management techniques, rename and register file throttling, do not contribute to a large power savings while enabled, as these techniques are designed to target specific temperature hotspots and thus have very little impact on global power dissipation. However, from an energy-efficiency point of view, local techniques can be competitive because in some cases they offer better performance than global schemes.

Figure 5.8 shows the results for the ST machine. Because DTM is rarely engaged for the ST architecture, there is a relatively small power overhead for these benchmarks. These ST results provide a baseline to decide whether SMT and CMP are still energy-efficient after DTM techniques

are applied.

From Figure 5.9 we can see that the SMT architecture is superior to the ST architecture for DVS and register renaming in terms of ED^2. As expected, the DVS techniques perform quite well, although with high-L2 miss rate benchmarks register file throttling, due to performance advantages, does nearly as well as DVS for ED^2.

Figure 5.10 allows us to compare CMP to the ST and SMT machines for energy-efficiency after applying DTM. When comparing CMP and SMT, we see that for the low-L2 miss rate benchmarks, the CMP architecture is always superior to the SMT architecture for all DTM configurations. In general, the local DTM techniques do not perform as well for CMP as they did for SMT. We see the exact opposite behavior when considering high-L2 miss rate benchmarks. In looking at the comparison between SMT and CMP architectures, we see that for the high-L2 miss rate benchmarks, CMP is not energy-efficient relative to *either* the baseline ST machine or the SMT machine—even with the DVS thermal management technique.

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

5.6 Future Work and Conclusions

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

- CMP and SMT exhibit similar operating *temperatures* within current generation process technologies, but the heating *behaviors* are quite different. SMT heating is primarily caused by localized heating within certain key microarchitectural structures such as the register file, due

to increased utilization. CMP heating is primarily caused by the global impact of increased energy output.

- In future process technologies in which leakage power is a significant percentage of the overall chip power CMP machines will generally be hotter than SMT machines. For the SMT architecture, this is primarily due to the fact that the increased SMT utilization is overshadowed by additional leakage power. With the CMP machine, replacing the relatively cool L2 cache with a second core causes additional leakage power due to the temperature-dependent component of subthreshold leakage.

- For the organizations this work studies, CMP machines offer significantly more throughput than SMT machines for CPU-bound applications, and this leads to significant energy-efficiency savings despite a substantial (80%+) increase in power dissipation. However, in the equal-area comparisons between SMT and CMP, the loss of L2 cache hurts the performance of CMP for L2-bound applications, and SMT is able to exploit significant thread-level parallelism. From an energy standpoint, the CMP machine's additional performance is no longer able to make up for the increased power output and energy-efficiency becomes negative.

- CMP and SMT cores tend to perform better with different DTM techniques. In general, in performance-oriented systems, localized DTM techniques work better for SMT cores and global DTM techniques work better for CMP cores. For energy-oriented systems, global DVS thermal management techniques offer significant energy savings. However, the performance benefits of localized DTM make these techniques competitive for techniques for energy-oriented SMT machines.

Future work includes exploring the impact of varying core complexity on the performance of SMT and CMP, and exploring a wider range of design options, like SMT fetch policies. There is also significant opportunity to explore tradeoffs between exploiting TLP and core-level ILP from energy and thermal standpoints. Finally, it is worthwhile to explore server-oriented workloads

which are likely to contain characteristics that are most similar to the memory-bound benchmarks from this study.

Chapter 6

CMP Design Space Exploration

6.1 Introduction

Recent product announcements show a trend toward aggressive integration of multiple cores on a single chip to maximize throughput. However, this trend presents an expansive design space for chip architects, encompassing the number of cores per die, core size and complexity (pipeline depth and superscalar width), memory hierarchy design, operating voltage and frequency, and so forth. Identifying optimal designs is especially difficult because the variables of interest are inter-related and must be considered simultaneously. Furthermore, trade-offs among these design choices vary depending both on workloads and physical (e.g., area and thermal) constraints.

This chapter explores this multi-dimensional design space across a range of possible chip sizes and thermal constraints, for both CPU-bound and memory-bound workloads. Few prior works have considered so many cores, and to my knowledge, this is the first work to optimize across so many design variables simultaneously. This chapter shows the inter-related nature of these parameters and how the optimum choice of design parameters can shift dramatically depending on system constraints. Specifically, this work demonstrates that:

- A simple, fast approach to simulate a large number of cores by observing that cores only interact through the L2 cache and shared interconnect. This methodology uses single-core traces and only requires fast cache simulation for multi-core results.

- CPU- and memory-bound applications desire dramatically different configurations. Adaptivity helps, but any compromise incurs throughput penalties.
- Thermal constraints dominate power-delivery constraints. Once thermal constraints have been met, throughput is throttled back sufficiently to meet current ITRS power-delivery constraints. Severe thermal constraints can even dominate pin-bandwidth constraints.
- A design must be optimized with thermal constraints. Scaling from the thermal-blind optimum leads to a configuration that is inferior, sometimes radically so, to a thermally optimized configuration.
- Simpler, smaller cores are preferred under some constraints. In thermally constrained designs, the main determinant is not simply maximizing the number of cores, but maximizing their power efficiency. Thermal constraints generally favor shallower pipelines and lower clock frequencies.
- Additional cores increase throughput, despite the resulting voltage and frequency scaling required to meet thermal constraints, until performance gains from an additional core is negated by the impact of voltage and frequency scaling across all cores.
- For aggressive cooling solutions, reducing power density is at least as important as reducing total power. For low-cost cooling solutions, however, reducing total power is more important.

This chapter is organized as follows. Section 6.2 is the related work. Section 6.3 introduces the model infrastructure and validation methodology. Section 6.4 presents design space exploration results and explanations. This chapter ends with conclusions and proposals for future work in section 6.5. This work is also published in [53].

6.2 Related Work

There has been a burst of work in recent years to understand the performance, energy, and thermal efficiency of different CMP organizations. Few have looked at a large numbers of cores and none, at

the time this work is published, have jointly optimized across the large number of design parameters this work considers while addressing the associated methodology challenges. Li and Martínez [49] present the most aggressive study of which the author is aware, exploring up to 16-way CMPs for SPLASH benchmarks and considering power constraints. Their results show that parallel execution on a CMP can improve energy efficiency compared to the same performance achieved via single-threaded execution, and that even within the power budget of a single core, a CMP allows substantial speedups compared to single-threaded execution.

Kongetira et al. [38] describe the Sun Niagara processor, an eight-way CMP supporting four threads per core and targeting workloads with high degrees of thread-level parallelism. Chaudhry et al. [13] describe the benefits of multiple cores and multiple threads, sharing eight cores with a single L2 cache. They also describe the Sun Rock processor's "scouting" mechanism that uses a helper thread to prefetch instructions and data.

El-Moursy et al. [21] show the advantages of clustered architectures and evaluate a CMP of multi-threaded, multi-cluster cores with support for up to eight contexts. Huh et al. [32] categorized the SPEC benchmarks into CPU-bound, cache-sensitive, or bandwidth-limited groups and explored core complexity, area efficiency, and pin-bandwidth limitations, concluding due to pin-bandwidth limitations that a smaller number of high-performance cores maximizes throughput. Ekman and Stenstrom [20] use SPLASH benchmarks to explore a similar design space for energy-efficiency with the same conclusions.

Kumar et al. [45] consider the performance, power, and area impact of the interconnection network in CMP architecture. They advocate low degrees of sharing, but use transaction oriented workloads with high degrees of inter-thread sharing. Since this work is modeling throughput-oriented workloads consisting of independent threads, this work follows the example of Niagara [38] and employ more aggressive L2 sharing. In the experiments of this work, each L2 cache bank is shared by half the total number of cores. Interconnection design parameters are not variable in the design space exploration of this work, and in fact constitute a sufficiently expansive design space of their own.

The research presented in this chapter differs from prior work in the large number of design

parameters and metrics this work considers. This work evaluates CMP designs for performance, power efficiency, and thermal efficiency while varying the number of cores per chip, pipeline depth and width, chip thermal packaging effectiveness, chip area, and L2 cache size. This evaluation is performed with a fast decoupled simulation infrastructure that separates core simulation from interconnection/cache simulation. By considering many more parameters in the design space, this work demonstrates the effectiveness of this infrastructure and show the inter-relatedness of these parameters.

The methodologies for analyzing pipeline depth and width build on prior work by Lee and Brooks [47] by developing first-order models for capturing changes in core area as pipeline dimensions change, thereby enabling power density and temperature analysis. This work identifies optimal pipeline dimensions in the context of CMP architectures, whereas most prior pipeline analysis considers single-core microprocessors [25,28,72], furthermore, most prior work in optimizing pipelines focused exclusively on performance, although Zyuban et al. found 18FO4 delays to be power-performance optimal for a single-threaded microprocessor [77].

Other researchers have proposed simplified processor models, with the goal of accelerating simulation. Within the microprocessor core, Karkhanis and Smith [36] describe a trace-driven, first-order modeling approach to estimate IPC by adjusting an ideal IPC to account for branch misprediction. In contrast, our methodology adjusts power, performance, and temperature estimates from detailed single-core simulations to account for fabric events, such as cache misses and bus contention. In order to model large scale multiprocessor systems running commercial workloads, Kunkel et al. [46] utilize an approach that combines functional simulation, hardware trace collection, and probabilistic queuing models. However, the decoupled and iterative approach allows this work to account for effects such as latency overlap due to out-of-order execution, effects not easily captured by queuing models. Although decoupled simulation frameworks have been proposed in the context of single-core simulation (e.g., Kumar and Davidson [40]) with arguments similar to this chapter's, the methodology used in this work is applied in the context of simulating multi-core processors.

6.3 Experimental Methodology

To facilitate the exploration of large CMP design spaces, This work proposes decoupling core and interconnect/cache simulation to reduce simulation time. Detailed, cycle-accurate simulations of multi-core organizations are expensive, and the multi-dimensional search of the design space, even with just homogeneous cores, is prohibitive. Decoupling core and interconnect/cache simulation dramatically reduces simulation cost with minimal loss in accuracy. The Turandot simulator is used to generate single-core L2 cache-access traces that are annotated with timestamps and power values. These traces are then fed to Zauber, a cache simulator that is developed in this work to model the interaction of multiple threads on one or more shared interconnects and one or more L2 caches. Zauber uses hits and misses to shift the time and power values in the original traces. Generating the traces is therefore a one-time cost, while what would otherwise be a costly multiprocessor simulation is reduced to a much faster cache simulation. Using Zauber, it is cost-effective to search the entire multi-core design space.

6.3.1 Simulator Infrastructure

The framework in this work decouples core and interconnect/cache simulation to reduce simulation time. Detailed core simulation provides performance and power data for various core designs, while interconnect/cache simulation projects the impact of core interaction on these metrics.

6.3.1.1 Core Simulation

Turandot and PowerTimer are extended to model the performance and power as pipeline depth and width vary using techniques from prior work [47].

Depth Performance Scaling: Pipeline depth is quantified in terms of FO4 delays per pipeline stage.[1] The performance model for architectures with varying pipeline depths are derived from the reference 19FO4 design by treating the total number of logic levels as constant and independent of the number of pipeline stages. This is an abstraction for the purpose of the analysis; increasing the

[1]Fan-out-of-four (FO4) delay is defined as the delay of one inverter driving four copies of an equally sized inverter. When logic and overhead per pipeline stage is measured in terms of FO4 delay, deeper pipelines have smaller FO4 delays.

Fetch		Decode	
NFA Predictor	1	Multiple Decode	2
L2 I-Cache	11	Millicode Decode	2
L3 I-Load	8	Expand String	2
I-TLB Miss	10	Mispredict Cycles	3
L2 I-TLB Miss	50	Register Read	1
Execution		**Memory**	
Fix Execute	1	L1 D-Load	3
Float Execute	4	L2 D-Load	9
Branch Execute	1	L3 D-Load	77
Float Divide	12	Float Load	2
Integer Multiply	7	D-TLB Miss	7
Integer Divide	35	L2 D-TLB Miss	50
Retire Delay	2	StoreQ Forward	4

Table 6.1: **Latencies for 19FO4 (cycles) [53]**

pipeline depth could require logic design changes. The baseline latencies (Table 6.1) are scaled to account for pipeline depth changes according to Eq. (6.1). These scaled latencies account for latch delays ($FO4_{latch} = 3$) and all latencies have a minimum of one cycle. This is consistent with prior work in pipeline depth simulation and analysis for a single-threaded core [77].

$$Lat_{target} = \left\lfloor Lat_{base} \times \frac{FO4_{base} - FO4_{latch}}{FO4_{target} - FO4_{latch}} + 0.5 \right\rfloor \tag{6.1}$$

Depth Power Scaling: Each factor in the standard equation for dynamic power dissipation, Eq. (6.2), scales with pipeline depth. The clock frequency f increases linearly with depth as the delay for each pipeline stage decreases. The clock gating factor CGF decreases by a workload dependent factor as pipeline depth increases due to the increased number of cycles in which the shorter pipeline stages are stalled. As the true switching factor α is independent of the pipeline depth and the glitching factor β decreases with pipeline depth due to shorter distances between latches, switching power dissipation decreases with pipeline depth. The latch count, and consequently hold power dissipation, increases linearly with pipeline depth. A detailed treatment of these scaling models can be found in [77].

	8D	4D	2D	1D
Functional Units				
FXU	4	2	1	1
MEM	4	2	1	1
FXU	4	2	1	1
BR	4	2	1	1
CR	2	1	1	1
Pipeline Stage Widths				
FETCH	16	8	4	2
DECODE	8	4	2	1
RENAME	8	4	2	1
DISPATCH	8	4	2	1
RETIRE	8	4	2	1

Table 6.2: **Resource Sizes with Width Scaling [53]**

$$P_{dyn} = CV^2 f(\alpha + \beta) \times CGF \tag{6.2}$$

Width Performance Scaling: The pipeline width is quantified in terms of the maximum number of instructions decoded per cycle. Performance data for architectures with varying pipeline widths are obtained from the reference 4-decode design (4D) by a linear scaling of the number of functional units and the number of non-branch instructions fetched, decoded, renamed, dispatched, and retired per cycle (Table 6.2). All pipelines have at least one instance of each functional unit. As pipeline width decreases, the number of instances of each functional unit is quickly minimized to one. Thus, the decode width becomes the constraining parameter for instruction throughput for the narrower pipelines this work considers (e.g., 2D).

Width Power Scaling: A hybrid approach is employed to model the power impact of scaling the width of the pipeline. The baseline microarchitecture, based on the POWER4, includes a clustered backend microarchitecture for structures like the functional units, issue queues, and register files. This approach is effective at managing complexity, cycle time, and power dissipation in wide-

Structure	Energy Growth Factor
Register Rename	1.1
Instruction Issue	1.9
Memory Unit	1.5
Multi-ported Register File	1.8
Data Bypass	1.6
Functional Units	1.0

Table 6.3: **Energy Scaling [53]**

issue superscalar cores [12, 63, 79]. An analogous technique is used to construct the dual-ported data cache. When scaling the width of these structures, this work assumes that unconstrained hold and switching power increases linearly with the number of functional units, access ports, and any other parameter that must change as width varies.

In certain non-clustered structures, however, linear power scaling may be inaccurate and, for example, does not capture non-linear relationships between power and the number of SRAM access ports since it does not account for the additional circuitry required in a multi-ported SRAM cell. For this reason, superlinear power scaling is applied with exponents (Table 6.3) drawn from Zyuban's work in estimating energy growth parameters [79]. Since these parameters were experimentally derived through analysis of non-clustered architecture, this work only applies this power scaling to the non-clustered components of the assumed architecture.

6.3.1.2 Interconnection/Cache Simulation

The core simulators are supplemented by Zauber, a much faster simulator that performs interpolation on L2 cache traces provided by the core simulators. Zauber decouples detailed core simulation and the simulation of core interaction. The cores in a CMP architecture usually share one or more L2 caches through an interconnection fabric. Therefore, resource contention between cores occurs primarily in these two resources. It is possible to simulate cache and fabric contention independent of core simulations without losing too much accuracy. The impact of contention on the performance and power of each core may then be evaluated quickly using interpolation.

First, L2 access traces are collected based on L1 cache misses through one pass of single-core simulations with a specific L2 cache size (0.5MB in this experiment). This work found these L2 traces to be independent of the L2 cache size. In these traces the L2 cache address and access time (denoted by the cycle) information is recorded for every access. Zauber also needs to sweep through a range of L2 cache sizes for each benchmark and record the performance and microarchitectural resource utilization every 10k instructions as this information will be used in the interpolation. These L2 traces are fed into an cache simulator and interconnection-contention model that reads the L2 accesses of each core from the traces, sorts them according to time of access, and uses them to drive the interconnection and L2 cache simulation. This interconnection/cache simulator outputs the L2 miss ratio and the delay due to contention for every 10k instruction segment of the thread running on each core.

With this L2 miss ratio and interconnection contention information we can calculate the new performance and power number for each 10k instruction segment of all the threads. Since we know the performance and microarchitectural resource utilization for several L2 miss ratio values, we are able to obtain new performance and utilization data for any other L2 miss ratio produced by the cache simulator via interpolation. Power numbers can be derived from the structure utilization data with post-processing.

When Zauber interleaves the L2 accesses from each thread, it is using the cycle information attached with each access to sort them by time of access. However, each thread may suffer different degrees of performance degradation due to interconnection and L2 cache contention. Therefore, sorting by time of access may not reflect the real ordering. The model in this work iterates to improve accuracy. In particular, given the performance impact from cache contention for each thread, Zauber can use this information to adjust the time of each L2 access in each L2 trace and redo L2 cache emulation based on this new L2 access timing information. Iterating to convergence, it is found three iterations are typically enough to reach good accuracy.

This work validates Zauber against the detailed cycle-accurate simulator, Turandot. Figure 6.1 shows the average performance and power data from Turandot simulation and Zauber simulation for 2-way and 4-way CMPs. From these figures, the average performance and power difference

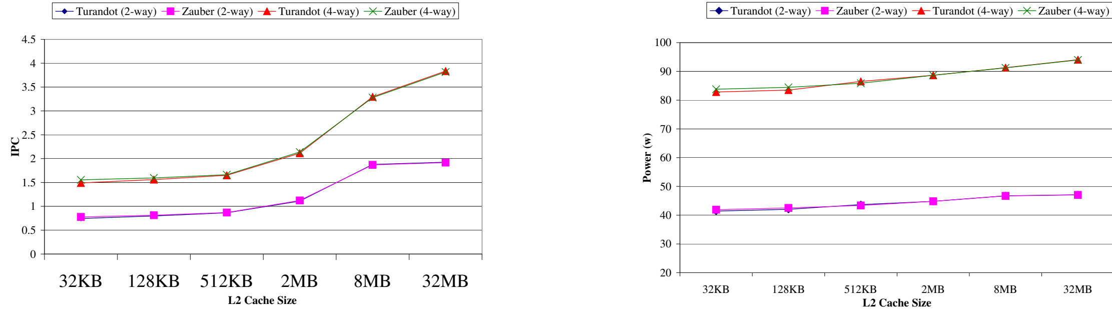


Figure 6.1: **The validation of Zauber model [53]**

between Turandot and Zauber is within 1%. For a 2-way CMP, Zauber achieves a simulation time speedup of 40-60x, with detailed Turandot simulations requiring 1-2 hours and the decoupled simulator requiring 1-3 minutes.

Since this work is modeling throughput-oriented workloads consisting of independent threads, a relatively high degree of cache sharing like Niagara [38] is assumed. Each L2 cache bank is shared by half the total number of cores. The interconnection power overheads are extrapolated from [45].

This work assumes the L2 cache latency does not change when the L2 cache size is varied. It also omits the effects of clock propagation on chip throughput and power when core number increases.

6.3.2 Analytical Infrastructure

This work uses formulas to vary and calculate parameters of interest in the CMP design space exploration. The design parameters this work considers include core count, core pipeline dimensions, thermal resistance of chip packaging, and L2 cache size. As these parameters are varied, this work considers the impact on both power and performance metrics.

6.3.2.1 Performance and Power Modeling

The analytical model uses performance and dynamic power data generated by Zauber simulation. Leakage power density for a given technology is calculated by Eq. (6.3), where A and B are coefficients determined by a linear regression of ITRS data and T is the absolute temperature. $A = 207.94$ and $B = 1446$ for 65nm technology.

$$P_{\text{leakage density}} = A \cdot T^2 \cdot e^{-B/T} \quad (6.3)$$

6.3.2.2 Temperature Modeling

This work uses steady-state temperature at the granularity of each core to estimate the chip thermal effects. This neglects localized hotspots within a core as well as lateral thermal coupling among cores. Addressing these is important future work, but employing a simple analytical temperature formula instead of the more complex models in HotSpot reduces simulation time and allows us to focus on how heat-removal limitations constrain core count and core type.

It is observed that the heat spreader is almost isothermal for the range of the chip areas and power values that are investigated, so we can separate the global temperature rise across the thermal package due to total chip power dissipation from localized temperature rise above the package due to per-core power dissipation. This is described by Equations 6.4, 6.5, and 6.6. Suppose we want to calculate the temperature of a specific core on the chip, where P_{glo} and P_{core} are the global chip and single core dynamic power, respectively. Similarly, L_{glo} and L_{core} are the global chip and single core leakage power, respectively. The chip's total dynamic power is the sum of the dynamic power dissipated by all the cores on chip, the L2 cache and the interconnect. The chip leakage power is summed in a similar manner. The sum of R_{spread} and R_{hsink} denotes the thermal resistance from the heat spreader to the air and the sum of $R_{silicon}$ and R_{TIM} denotes the thermal resistance from the core. Collectively, these parameters specify the chip's thermal characteristics from the device level to the heat spreader, ignoring the lateral thermal coupling above the heat spreader level.

This work categorizes the CMP heatup into local and global effects. The former is determined by the local power dissipation of any given core and the effect on its temperature. The latter is determined by the global chip power.

$$
\begin{aligned}
H_{glo} + H_{loc} &= T_{core} - T_{amb} && (6.4) \\
H_{glo} &= (P_{glo} + L_{glo}) \cdot (R_{spread} + R_{hsink}) && (6.5) \\
H_{loc} &= (P_{core} + L_{core}) \cdot (R_{silicon} + R_{TIM}) && (6.6)
\end{aligned}
$$

This distinction between local and global heatup mechanisms is first qualitatively introduced by us in [54]. We observed that adding cores to a chip of fixed area increases chip temperature. This observation may not be evident strictly from the perspective of per-core power density. Although power density is often used as a proxy for steady-state temperature with each core exhibiting the same power density, core or unit power density is only an accurate predictor of the temperature increases in the silicon relative to the package. Per-unit or per-core power density is analogous to one of the many thermal resistances comprising the entire network that represents the chip.

Adding cores does indeed increase temperature, because it increases the total amount of power that must be removed. The current primary heat removal path is convection from a heat sink. Although accurate expressions for convective heat transfer are complex, a first-order approximation is:

$$q = hA(T_{sink} - T_{air}) \quad (6.7)$$

where q is the rate of convective heat transfer, h is the convective heat transfer coefficient that incorporates air speed and various airflow properties, A is the surface area for convection, and to first order T_{air} can be assumed to be fixed. At steady-state, the total rate of heat P generated in the chip must equal the total rate of heat removed from the chip. If h and A are held constant, then as cores are added and P exceeds q, T_{sink} must rise to balance the rates so that $P = q$. This increases

on-chip temperatures because the sink temperature is like an offset for the other layers from the sink-spreader interface through the chip.

Alternative heat removal mechanisms also warrant consideration. For example, fan speed may be increased, but this approach is often limited by acoustical limits and various board-layout and airflow factors that lead to diminishing returns (e.g. increased pressure drop across a larger heat sink). We can lower the inlet air temperature, but this is not an option in many operating environments (e.g. a home office), or may be extremely costly (e.g. in a large data center). We could also increase heat sink area, but this is where Eq. (6.7) breaks down. That expression assumes that the heat source is similar in size to the conductive surface. In reality, increasing the heat sink surface area does not improve convective heat transfer in a simple way. Increasing fin height and surface area is limited by airflow constraints that dictate an optimal fin configuration. Increasing the total size of the heat sink (i.e. increasing the area of its base), leads to diminishing returns as the ratio of sink to chip area increases due to limitations on how well the heat can be spread. In the limit, the heat source looks like a point source and further increases in the sink area will have no benefit, as heat will not be able to spread at all to the outermost regions of the heat sink.

With regard to the single thermal resistance in the HotSpot model, adding cores is equivalent to adding current sources connected in parallel to a single resistor, the sink-to-air resistance. The increased current leads to a larger IR drop across this resistor and a proportionally larger heat-sink temperature.

Equations 6.4, 6.5, and 6.6, quantify the contributions from global and local heatup. Figure 6.2 presents results from validating this simple model against HotSpot, varying the heat sink resistance and fixing the power distribution to test different temperature ranges. The temperature difference between these two models is normally within 3°.

6.3.2.3 Area Modeling

This work assumes a 65nm technology. Based on a Power5 die photo, the baseline core area is estimated to be $11.52mm^2$, equivalent to the area of 1MB of L2 cache. This work assumes each $n/2$ cores share one L2 cache through a crossbar routing over the L2 and estimate the total crossbar area

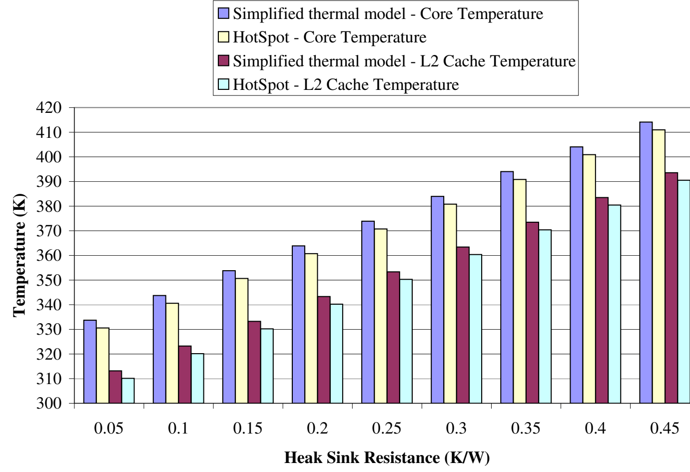


Figure 6.2: **Simplified temperature model validation [53]**

to be $5.7n \cdot mm^2$ [45], where n is the number of cores. As pipeline dimensions vary, the core area is scaled to account for additional structures and overhead.

Depth Area Scaling: Given the assumption of fixed logic area independent of pipeline depth, latch area constitutes the primary change in core area as depth varies. Let w_{latch} be the total channel width of all pipeline latch transistors, including local clock distribution circuitry. Let w_{total} be the total channel width for all transistors in the baseline microprocessor, excluding all low-leakage transistors in on-chip memories. Let the latch growth factor (LGF) capture the latch count growth due to logic shape functions. This analysis sets the latch ratio (w_{latch}/w_{total}) at 0.3 and the LGF at 1.1, assuming superlinear latch growth as pipeline depth increases [77]. Assuming changes in core area are proportional to the total channel width of latch transistors in the pipeline, the portion of core area attributed to latches is scaled superlinearly with pipeline depth using Eq. (6.8).

$$A_{target} = A_{base}\left(1 + \frac{w_{latch}}{w_{total}}\left(\left(\frac{FO4_{target}}{FO4_{base}}\right)^{LGF} - 1\right)\right) \tag{6.8}$$

Width Area Scaling: Table 6.4 presents area scaling factors for varying pipeline width. This work considers each unit and its underlying macros. To first-order, the core area attributed to the

Unit/Macro	2D	4D	8D
FXU	0.5	1.0	2.0
FPU	0.5	1.0	2.0
ISU	0.6	1.0	1.8
IFU	1.0	1.0	1.0
LSU	0.5	1.0	2.0
IDU	1.0	1.0	1.0
Total	**0.7**	**1.0**	**1.7**

Table 6.4: **Pipeline Width Area Scaling [53]**

fixed point, floating point, and load store units scale linearly due to clustering. It is also assumed that the area of multi-ported SRAM array structures is wire dominated and scales linearly with the number of ports [74, 78]. This assumption applies to SRAM memories (*e.g.* register files), but may be extended to queues (*e.g.* issue queues), tables (*e.g.* rename mappers), and other structures potentially implemented as an SRAM array.

Note the area of the instruction fetch and decode units (IFU, IDU) are independent of width. Within the fetch unit, the instruction cache, instruction TLB, program counter, and branch handling hardware dominate the fetch unit's total power dissipation and, to first-order, these structures are independent of the fetch width. Within the decode unit, the instruction decoder ROM used to crack complex instructions dominates decode power and, to first-order, this ROM is independent of decode width. Also note that only a subset of the macros for the instruction sequencing unit (ISU) scale as width increases, resulting in a sublinear area dependence on width for this unit. For the sequencing unit, only area associated with issue queues and tables for register renaming scale with pipeline width. The total scaling factors, a weighted average of the unit scaling factors, suggest a sublinear relationship between area and width.

6.3.2.4 DVFS Scaling and Reward Functions

Using a large number of cores may lead to thermal run-away due to high chip power and the positive feedback of leakage power and temperature. A thermal control mechanism must be employed to prevent this behavior and to account for the resulting performance impact. This work takes this

control into consideration by emulating voltage and frequency scaling for steady-state temperature control. The dynamic simulations here do not model the dynamic control aspect of DVFS. Instead, this work only simulates workloads in which all cores are occupied — "worst typical-case" workloads that are likely to dictate thermal design. Then, for a given workload, we can calculate its steady-state temperature and infer the voltage and frequency settings necessary to prevent the steady-state temperature from pushing the chip above 100°. These settings could represent the maximum steady-state or nominal settings that are safe for "worst typical-case" workloads, or could represent steady-state V/f values with DVFS when these workloads are running. In reality, the DVFS settings would fluctuate around these values with such workloads, permitting higher settings when fewer cores are occupied.

$$\begin{aligned} T_{thr} - T_{amb} &= (P_{global}R_{below} + P_{core}R_{above})V_{sc}^2 F_{sc} \\ &\quad + (L_{global}R_{below} + L_{core}R_{above})V_{sc} \end{aligned} \tag{6.9}$$

For a given core number n, L2 cache size l, pipeline depth d, and pipeline width w, the dynamic power consumption and performance are obtained from Zauber and the leakage power with Equation 6.3. For a given temperature threshold, the voltage and frequency scaling factors are calculated from Equation 6.9, which is deduced from Equation 6.4, assuming that the leakage power is mainly subthreshold leakage power and is linearly dependent on voltage. Using 0.9V and 2.0 GHz as the nominal voltage and clock frequency and their scaling factors as V_{sc} and F_{sc}, this work uses a nonlinear voltage/frequency relationship obtained from HSPICE circuit simulation. After determining the voltage and frequency scaling required for thermal control, the reward functions, BIPS and BIPS^3/W, are calculated with Equations 6.10 and 6.11.

$$BIPS(n,l,d,w) = BIPS_{base} \cdot F_{sc} \tag{6.10}$$

$$\frac{BIPS^3}{W}(n,l,d,w) = \left(\frac{{BIPS_{base}}^3}{P_{dyn} + \frac{P_{leak}}{V_{sc}F_{sc}}} \right) \left(\frac{F_{sc}}{V_{sc}} \right)^2 \tag{6.11}$$

6.3.3 Workloads

This work characterizes all SPEC2000 benchmarks into eight major categories: high IPC(> 0.9) or low IPC(< 0.9), high temperature(peak temperature $> 355K$) or low temperature(peak temperature $< 355K$), floating-point or integer benchmark. Eight of the SPEC2000 benchmarks (art, mcf, applu, crafty, gcc, eon, mgrid, swim) are employed as single thread benchmarks, spanning these categories. This work further categorizes benchmarks according to their L2 miss ratios, referring to those with high and low miss ratios as memory- and CPU- bound, respectively.

To generate static traces, this work compiles with the *xlc* compiler and -O3 option. Simpoint [23] is used to identify representative simulation points and generate traces by capturing 100 million instructions beginning at the Simpoint.

For both CPU-bound and memory-bound benchmarks, pairs of single-thread benchmarks are used to form dual-thread benchmarks and replicate these pairs to form multiple benchmark groups of each benchmark category for CMP simulation with more than two cores. This work only simulates workloads consisting of a large pool of waiting threads to keep all cores active, representing the "worst typical-case" operation likely to determine physical limits.

6.4 Results

This section presents the results from the exploration of a large CMP design space that encompasses core count, pipeline dimensions, and cache size. This research considers optimizing for performance (BIPS) and power-performance efficiency (BIPS^3/W) under various area and thermal constraints. In addition to demonstrating the effectiveness of the experimental methodology for exploring large design spaces, the results also quantify significant CMP design trends and demonstrate the need to make balanced design choices.

6.4.1 Optimal Configurations

Table 6.5 and Table 6.6 present optimal configurations that maximize BIPS and BIPS^3/W for a fixed pipeline depth while Table 6.7 and Table 6.8 present optima for a fixed superscalar width.

	L2 (MB)	Core Number	Pipeline Width	Voltage Scaling	Frequency Scaling
nolimit+NT+CPU	32	20	8	1.00	1.00
nolimit+LR+CPU	8	20	4	0.75	0.63
nolimit+HR+CPU	2	18	2	0.59	0.39
400+NT+CPU	4	20	4	1.00	1.00
400+LR+CPU	4	20	4	0.75	0.64
400+HR+CPU	2	18	2	0.59	0.39
200+NT+CPU	2	10	4	1.00	1.00
200+LR+CPU	2	10	4	0.87	0.80
200+HR+CPU	2	12	2	0.67	0.51
100+NT+CPU	2	4	4	1.00	1.00
100+LR+CPU	2	4	4	0.97	0.96
100+HR+CPU	2	4	4	0.79	0.70
nolimit+NT+MEMORY	32	20	8	1.00	1.00
nolimit+LR+MEMORY	16	20	4	0.73	0.61
nolimit+HR+MEMORY	8	10	2	0.62	0.45
400+NT+MEMORY	16	16	2	1.00	1.00
400+LR+MEMORY	16	12	4	0.81	0.73
400+HR+MEMORY	8	10	2	0.62	0.45
200+NT+MEMORY	8	8	2	1.00	1.00
200+LR+MEMORY	8	6	4	0.93	0.90
200+HR+MEMORY	8	8	2	0.66	0.51
100+NT+MEMORY	2	4	4	1.00	1.00
100+LR+MEMORY	2	4	4	1.00	1.00
100+HR+MEMORY	2	4	4	0.81	0.73

Table 6.5: **Optimal Configurations for BIPS with Varying Pipeline Width, Fixed Depth (18FO4) [53]**

Configurations are presented for various combinations of area and thermal constraints. The area constraint can take on one of four values: no constraint ("nolimit"), $100mm^2$, $200mm^2$, or $400mm^2$. Similarly, packaging assumptions and hence thermal constraints can take on one of three values: no constraint (NT), low constraint (LR=0.1, low thermal resistance, i.e. aggressive , high-cost thermal solution), and high constraint (HR=0.45, high thermal resistance, i.e. constrained thermal solution, such as found in a laptop). The tables differentiate between CPU- and memory-bound benchmarks and specify the required voltage and frequency ratios needed to satisfy thermal constraints.

Figures 6.3–6.5 present performance trade-offs between core count, L2 cache size, and pipeline dimensions for a $400mm^2$ chip subject to various thermal constraints.

6.4.1.1 No Constraints

In the absence of area and thermal constraints (nolimit+NT+CPU, nolimit+NT+MEMORY), the throughput maximizing configuration for both CPU- and memory-bound benchmarks employs the largest L2 cache and number of cores. Although the optimal pipeline width for all benchmarks is

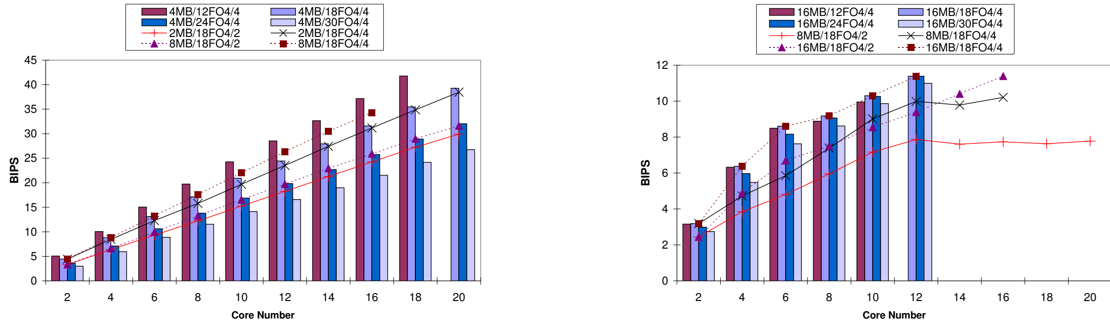


Figure 6.3: **Performance of various configurations with chip area constraint at 400mmˆ2 (without thermal control) [53]**

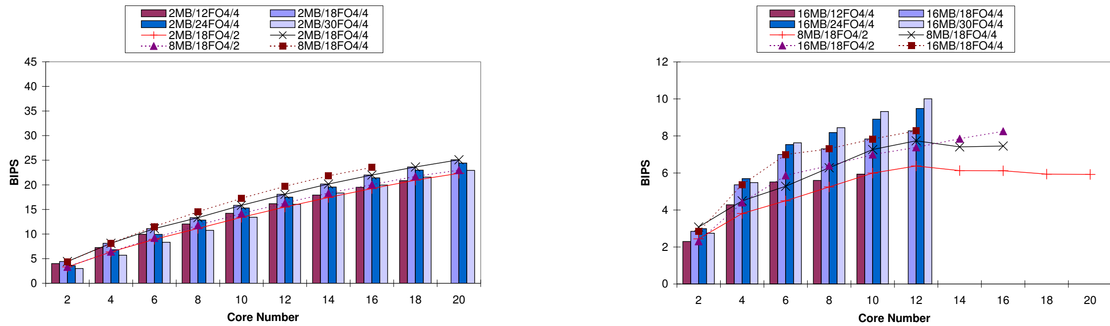


Figure 6.4: **Performance of various configurations with chip area constraint at 400mmˆ2 (R = 0.1 heat sink) [53]**

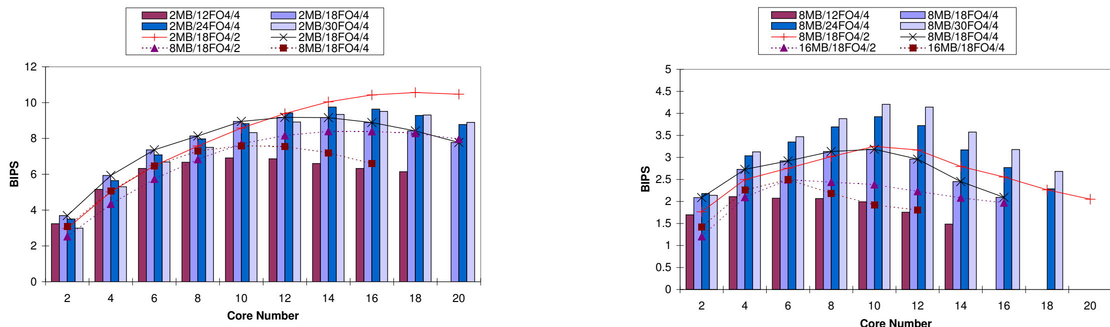


Figure 6.5: **Performance of various configurations with chip area constraint at 400mmˆ2 (R = 0.45 heat sink) [53]**

	L2 (MB)	Core Number	Pipeline Width	Voltage Scaling	Frequency Scaling
nolimit+NT+CPU	16	20	8	1.00	1.00
nolimit+LR+CPU	8	20	4	0.75	0.63
nolimit+HR+CPU	2	16	2	0.61	0.43
400+NT+CPU	4	20	4	1.00	1.00
400+LR+CPU	2	20	4	0.76	0.65
400+HR+CPU	2	16	2	0.61	0.43
200+NT+CPU	2	10	4	1.00	1.00
200+LR+CPU	2	12	2	0.90	0.85
200+HR+CPU	2	12	2	0.67	0.51
100+NT+CPU	2	4	4	1.00	1.00
100+LR+CPU	2	4	4	0.97	0.96
100+HR+CPU	2	4	4	0.79	0.70
nolimit+NT+MEMORY	32	20	4	1.00	1.00
nolimit+LR+MEMORY	16	20	4	0.73	0.61
nolimit+HR+MEMORY	8	10	2	0.62	0.45
400+NT+MEMORY	16	16	2	1.00	1.00
400+LR+MEMORY	16	16	2	0.81	0.72
400+HR+MEMORY	8	10	2	0.62	0.45
200+NT+MEMORY	8	8	2	1.00	1.00
200+LR+MEMORY	8	8	2	0.92	0.88
200+HR+MEMORY	8	8	2	0.66	0.51
100+NT+MEMORY	4	4	2	1.00	1.00
100+LR+MEMORY	4	4	2	0.98	0.98
100+HR+MEMORY	4	4	2	0.81	0.73

Table 6.6: **Optimal Configurations for BIPS^3/W with Varying Pipeline Width, Fixed Depth (18FO4) [53]**

eight (8W), CPU-bound benchmarks favor deeper pipelines (12FO4) to take advantage of fewer memory stalls and higher instruction level parallelism. Conversely, memory-bound benchmarks favor relatively shallow pipelines (18FO4).

For BIPS^3/W, the optimal depth shifts to shallower pipelines; 18FO4 and 30FO4 delays per stage are optimal for CPU and memory-bound benchmarks, respectively. The optimal width shifts to shallower, narrower pipelines for memory-bound benchmarks due to the relatively high rate of memory stalls and low instruction level parallelism.

6.4.1.2 Area Constraints

Considering area constraints ({100,200,400}+NT+*), we can find core number and L2 cache size tend to decrease as area constraints are imposed. Although both techniques are applied in certain cases (100+NT+CPU, 100+NT+MEMORY), decreasing the cache size is naturally the most effective approach to meet area constraints for CPU-bound benchmarks, while decreasing the number of

	L2 (MB)	Core Number	Pipeline Depth	Voltage Scaling	Frequency Scaling
nolimit+NT+CPU	32	20	12	1.00	1.00
nolimit+LR+CPU	8	20	18	0.75	0.63
nolimit+HR+CPU	2	14	24	0.62	0.44
400+NT+CPU	4	18	12	1.00	1.00
400+LR+CPU	4	20	18	0.75	0.64
400+HR+CPU	2	14	24	0.62	0.44
200+NT+CPU	2	10	18	1.00	1.00
200+LR+CPU	2	10	18	0.87	0.80
200+HR+CPU	2	10	18	0.63	0.45
100+NT+CPU	2	4	18	1.00	1.00
100+LR+CPU	2	4	18	0.97	0.96
100+HR+CPU	2	4	18	0.79	0.70
nolimit+NT+MEMORY	32	20	18	1.00	1.00
nolimit+LR+MEMORY	16	20	30	0.85	0.78
nolimit+HR+MEMORY	8	10	30	0.65	0.48
400+NT+MEMORY	16	12	18	1.00	1.00
400+LR+MEMORY	16	12	30	0.94	0.91
400+HR+MEMORY	8	10	30	0.65	0.48
200+NT+MEMORY	8	6	24	1.00	1.00
200+LR+MEMORY	8	6	24	1.00	1.00
200+HR+MEMORY	4	6	30	0.83	0.75
100+NT+MEMORY	2	4	24	1.00	1.00
100+LR+MEMORY	2	4	24	1.00	1.00
100+HR+MEMORY	2	4	30	0.96	0.95

Table 6.7: **Optimal Configurations for BIPS with Varying Pipeline Depth, Fixed Width (4D) [53]**

	L2 (MB)	Core Number	Pipeline Depth	Voltage Scaling	Frequency Scaling
nolimit+NT+CPU	16	20	18	1.00	1.00
nolimit+LR+CPU	8	20	18	0.75	0.63
nolimit+HR+CPU	2	14	24	0.62	0.44
400+NT+CPU	4	20	18	1.00	1.00
400+LR+CPU	2	20	24	0.85	0.78
400+HR+CPU	2	14	24	0.62	0.44
200+NT+CPU	2	10	18	1.00	1.00
200+LR+CPU	2	10	24	0.97	0.95
200+HR+CPU	2	10	24	0.69	0.55
100+NT+CPU	2	4	18	1.00	1.00
100+LR+CPU	2	4	18	0.97	0.96
100+HR+CPU	2	4	18	0.79	0.70
nolimit+NT+MEMORY	32	20	30	1.00	1.00
nolimit+LR+MEMORY	16	20	30	0.85	0.78
nolimit+HR+MEMORY	8	10	30	0.65	0.48
400+NT+MEMORY	16	12	30	1.00	1.00
400+LR+MEMORY	8	12	30	1.00	1.00
400+HR+MEMORY	8	10	30	0.65	0.48
200+NT+MEMORY	8	6	30	1.00	1.00
200+LR+MEMORY	8	6	30	1.00	1.00
200+HR+MEMORY	4	6	30	0.83	0.75
100+NT+MEMORY	2	4	30	1.00	1.00
100+LR+MEMORY	2	4	30	1.00	1.00
100+HR+MEMORY	2	4	30	0.96	0.95

Table 6.8: **Optimal Configurations for $BIPS^3/W$ with Varying Pipeline Depth, Fixed Width (4D) [53]**

cores is most effective for memory-bound benchmarks.

With regard to pipeline dimensions, the optimal width decreases to 2W for all area constraints on memory-bound benchmarks (*+NT+MEMORY) except 100+NT+MEMORY. According to the area models in Section 6.3.2.3, changes in depth scale the latch area (only 30% of total area) whereas changes in width scale the area associated with functional units, queues, and other width-sensitive structures. Thus, shifting to shallower widths provides greater area impact (Table 6.5). Although pipeline depths may shift from 12 to 18/24FO4 delays per stage, they are never reduced to 30FO4 delays per stage to meet area constraints (Table 6.7).

As in the case without constraints, the bar plots in Figure 6.3, which vary pipeline depth, shows CPU-bound benchmarks favor deeper pipelines (4MB/12FO4/4 is optimal) and memory-bound benchmarks favor shallower pipelines (16MB/18FO4/4 or 16MB/24FO4/4 are optimal). The line plots in Figures 6.3–6.4 also present performance for varying widths for modest thermal constraints. In this case, the optimal pipeline width is 4W for a fixed depth of 18FO4 delays per stage.

6.4.1.3 Thermal Constraints

We find thermal constraints (nolimit+{NT,LR,HR}+*), also shift optimal configurations to fewer and simpler cores. The optimal core number and L2 size tends to decrease with heat sink effectiveness. For example, the optimum for nolimit+HR+MEMORY is 8MB L2 cache and 10 cores. Again, CPU-bound benchmarks favor decreasing cache size to meet thermal constraints while memory-bound benchmarks favor decreasing the number of cores.

Figure 6.5 also illustrates the impact of global heating on optimal pipeline configurations. As the number of cores increase for CPU-bound benchmarks, the optimal delay per stage increases by 6FO4 (i.e., from 18 to 24FO4) when twelve cores reside on a single chip. The increasing core count increases chip temperature, leading to shallower pipelines that lower power dissipation, lower global temperature, and meet thermal constraints.

Simpler cores, characterized by smaller pipeline dimensions, tend to consume less power and, therefore, mitigate the core's thermal impact. In particular, the optimal pipeline depth shifts to 24 and 30FO4 delays per stage for CPU and memory-bound benchmarks, respectively, when compar-

ing nolimit+NT+* to nolimit+HR+* in Table 6.7. Similarly, the optimal width shifts to 2W for all benchmarks when comparing the same entries in Table 6.5.

Figures 6.4–6.5 show imposing thermal constraints shifts the optimal depth to shallower design points. The performance for CPU and memory-bound benchmarks are maximized for 18-24 and 24-30FO4 delays per stage, respectively. Pipeline power dissipation increases superlinearly with depth while pipeline area increases sublinearly according to Section 6.3.2.3. Thus, growth in power dissipation exceeds area growth and the overall power density increases with depth. Thus, optimal designs must shift to shallower pipelines to meet thermal constraints. Similarly, more aggressive thermal constraints, shown in Figure 6.5 shifts the optimal width to the narrower 2W, especially as the number of cores increases. These results also suggest thermal constraints will have a greater impact on pipeline configurations than area constraints.

6.4.1.4 Area and Thermal Comparison

Comparing the impact of thermal constraints (nolimit+NT+* versus nolimit+HR+*) to the impact of area constraints (nolimit+NT+* versus 100+NT+*) demonstrates larger shifts towards smaller pipeline dimensions. In general, thermal constraints exert a greater influence on the optimal design configurations.

Applying a more stringent area constraint reduces the trend towards simpler cores. With a smaller chip area, resulting in fewer cores and smaller caches, total power dissipated and the need for thermal control is diminished. As this occurs, pressure towards simpler cores with smaller pipeline dimensions also fades.

6.4.1.5 Depth and Width Comparison

Consider a baseline configuration 2MB/18FO4/4W. As thermal constraints are imposed, the configuration may either shift to a shallower core (2MB/24FO4/4W) or shift to a narrower core (2MB/18FO4/2W). Since changes in width scale area for both functional units and many queue structures, whereas changes in depth only scale area for latches between stages, width reductions have a greater area impact relative to depth reductions. Thus, the 2MB/24FO4/4W core is a larger

core relative to the 2MB/18FO4/2W and exhibits lower dynamic power density. However, the smaller 2MB/18FO4/2W core benefits from less leakage power per core and, consequently, less global power (since dynamic power dissipation is comparable for both cores).

From the temperature models in Section 6.3.2.2, total power output, P_{global}, has greater thermal impact for a chip with a poor heat sink (i.e., high thermal resistance, $R_{heatsink}$). Similarly, the thermal impact is dominated by the local power density, P_{core}, for a chip with a good heat sink. In this case, the transfer of heat from the silicon substrate to the spreader dominates thermal effects. Thus, to minimize chip heatup, it is advantageous to reduce width and global power in the context of a poor heat sink and advantageous to reduce depth and local power density in the context of a more expensive heat sink.

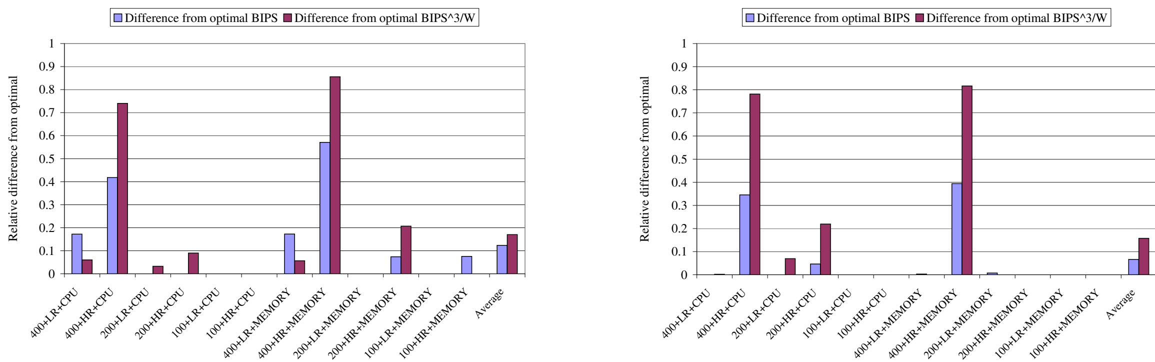


Figure 6.6: **The difference from the optimal when no thermal consideration is made at early design [53]**

6.4.2 Hazards of Neglecting Thermal Constraints

Thermal constraints should be considered early in the design process. If a chip is designed without thermal constraints in mind, designers must later cut voltage and clock frequency to meet thermal constraints. The resulting voltage and frequency, and hence performance, will likely be cut more severely than if a thermally-aware configuration were selected from the beginning. Figure 6.6 demonstrates the slowdown incurred by choosing a non-thermally optimal design with voltage and frequency scaling over the thermally-optimal design. The y-axis plots the thermal-aware optimal performance minus the performance of the configuration without thermal considerations,

normalized to the optimal performance. This figure summarizes the slowdown for all combinations of die sizes, heat-sink configurations, application classes, and for both pipeline depth and width optimizations. The average difference for varying depth is around 12-17% and 7-16% for varying width.

However, for large, $400mm^2$ chips, omitting thermal consideration may result in huge performance degradations. For example, the 400+HR+CPU and 400+HR+MEMORY configurations result in a 40% – 90% difference in performance for BIPS and $BIPS^3/W$. As area constraints are relaxed, the optimal point tends to include more cores and larger L2 caches. However, if the chip has severe thermal problems, DVFS scaling must scale aggressively to maintain thermal limits, into a region with significant non-linear voltage and frequency scaling, producing large performance losses. For smaller chips with fewer cores and smaller L2 caches, the difference may be negligible because there are very few configurations to choose from. As future CMP server-class microprocessors target $400mm^2$ chips with more than eight cores, it will be essential to perform thermal analysis in the early-stages of the design process when decisions about the number and complexity of cores are being performed.

6.4.3 DVFS Versus Core Sizing

In meeting thermal constraints for large CMP machines where global heat-up and total chip power is a concern, designers may be forced to choose among implementing fewer cores, smaller L2 caches, or employing aggressive DVFS scaling. This work finds DVFS superior to removing cores for CPU-bound applications as long as reductions in frequency are met by at least an equal reduction in dynamic and leakage power. Additional cores for CPU-bound applications provide linear increases in performance with near-linear increases in power dissipation. However, because of the strongly non-linear relationship between voltage scaling and clock frequency at low voltages, voltage scaling at some point stops providing super-linear power savings to make up for the performance (clock-frequency) loss. At this point, designers must consider removing cores and L2 cache from the design to meet thermal constraints.

For example, a chip with 30% leakage power no longer achieves super-linear power-

performance benefit from DVFS scaling after roughly 0.55x Vdd scaling; frequency of the chip drops to 0.18x and power dissipation also to 0.18x (dominated by leakage power, which only scales linearly with Vdd). Further reductions in Vdd lead to greater performance loss than power savings. (In future process technologies, more than 0.55x Vdd scaling may also approach reliability limits of conventional CMOS circuits.)

Figure 6.5 shows an example of this behavior with the 2MB/18FO4/4W design. When this design exceeds 14 cores, further increases in core count lead to performance degradation. Vdd scaling has exceeded 0.55x, and the additional DVFS scaling necessary to meet thermal constraints costs more performance than is gained by adding these additional cores. On the other hand, the 2MB/18FO4/2W design only requires Vdd scaling of 0.57x out to 20 cores, which is why this design is attractive even with the additional cores.

Similar analyses hold for memory-bound applications. In this case, the tradeoff is more complex, because the performance benefit from adding cores may be non-linear. In this case, designers must carefully tradeoff power-performance benefits of DVFS, L2, core complexity, and core sizing to maximize the overall performance benefit with the power costs.

6.4.4 Accommodating Heterogeneous Workloads

Figures 6.3–6.5 also highlight the difficulty of accommodating a range of workload types under area constraints.This is less of a concern when looking at a small number of cores like most prior studies. Prior studies have also neglected the role of pipeline dimensions, which we find to play a major role. And for large numbers of cores, radically different configurations are possible.

CPU-bound and memory-bound workloads have different, incompatible optima. The performance loss from using the CPU-bound optimum with the memory-bound workload and vice-versa is severe, 37–41% and 26–53% respectively, depending on thermal constraints. Even if we try to identify compromise configurations, it is surprising how poorly they perform for one or the other workload. Of course, the best compromise depends on how heavily each workload is weighted. This work tried to minimize the performance loss on both workloads.

With no thermal limits, the best configuration is 16 4-wide, 18FO4-deep cores with 8MB of

cache, incurring an 18% penalty for the CPU-bound workload. If we turn off 8 cores, it incurs 10% penalty for the memory-bound workload. Moving to 16MB improves memory-bound performance, but hurts CPU-bound performance because it sacrifices 8 cores with an area constraint of 400 mm^2.

With thermal limits, the optimal configurations begin to converge, as the maximum possible number of cores and the L2 cache size is constrained, as the BIPS benefit of extra cores is reduced for CPU-bound benchmarks, and as the benefit of additional cache lines is reduced for memory-bound benchmarks. For low thermal resistance, the best compromise is 18 4-wide cores and 8 MB. This incurs only a 4% performance loss for CPU-bound benchmark and a 10% loss for the memory-bound case. With high thermal resistance, the best compromise is 14 4-wide, 30FO4-deep cores with 8 MB of cache. Turning off 4 cores we reach the optimal configuration for memory-bound case, but this configuration incurs 12% penalty for the CPU-bound case.

Although the discrepancy between the needs of CPU- and memory-bound workloads narrows with increasing thermal constraints, some penalty seems inevitable, because CPU-bound benchmarks prefer more cores while memory-bound benchmarks prefer larger L2 caches. It is interesting to note that we do not see a simple heuristic for identifying good compromise configurations.

6.5 Future Work and Conclusions

The major conclusions include:

- Joint optimization across multiple design variables is necessary. Even pipeline depth, typically fixed in architecture studies, may impact core area and power enough to change the optimal core count. Optimizing without thermal constraints and then scaling to a thermal envelope leads to dramatically inferior designs compared to those obtained from including thermal constraints in the initial optimization.

- Thermal constraints appear to dominate other physical constraints like power delivery. Once thermal constraints are met, at least within the design space this work studied, power has been throttled sufficiently to fall safely within ITRS power-delivery projections.

- Thermal constraints tend to favor shallower pipelines and narrower cores, and tend to reduce the optimal number of cores and L2 cache size. Nevertheless, even under severe thermal constraints, additional cores benefit throughput despite aggressive reductions in operating voltage and frequency. This is true until performance gains from an additional core is negated by the impact of the additional voltage and frequency scaling required of all the cores. This inflection occurs at approximately 55% of the nominal Vdd, well into the range of non-linear frequency scaling (18% of nominal!).

- For aggressive cooling solutions, reducing power density is at least as important as reducing total power. For low-cost cooling solutions, however, reducing total power is more important because raising power dissipation (even if power density is the same) raises a chip's temperature.

These results raise a range of questions for future work, such as the need for adaptive chip architectures that can dynamically accommodate the full range of workloads, from heavily CPU-bound to heavily memory-bound. Examining how all findings here might change with other workloads (e.g., scientific parallel applications or communication-heavy commercial server workloads) and other architectures (e.g., in-order processors) is future work. Further research on L2/L3/Memory interconnect/hierarchy for CMP and on the impact of clock propagation on CMP throughput and power is also necessary.

While CMPs may optimize for throughput-oriented application workloads at the expense of single-thread performance, single-thread performance will still be an important consideration for many application domains. Addressing single-thread performance will likely require additional design tradeoffs. This does not necessarily require aggressive superscalar cores running at full voltage and frequency. Future research in this direction must consider speculative multithreading, heterogeneous cores, dynamic core adaptation, run-ahead execution/scouting, and so forth.

Chapter 7

The Investigation of Core Type Choices for CMP

7.1 Introduction

Recent years have seen a trend in the design of microprocessors towards chip multi-processors that couple shared on-chip caches with multiple CPU cores. These designs seek to provide increased energy-efficiency and excellent throughput performance. Sun's throughput computing initiative is an example of this direction. In this system architecture, a chip multiprocessor configuration with eight, simple in-order, heavily multi-threaded cores is employed. These systems provide an unparalleled number of computational threads within a very reasonable power envelope.

However, it is not clear that this approach provides the correct balance of throughput and latency for certain market segments. It is not just legacy applications that are sensitive to single-thread latency. Many Internet services will increasingly differentiate themselves on the basis of response time, possibly even guaranteeing minimum response times through service-level agreements. Some applications do not parallelize beyond a few cores, and even those that do may still be latency sensitive. This might occur when evaluating many scenarios, or when an application continuously refines an answer, to achieve the best result within a given latency constraint (as in some games, decision support, etc.).

Heterogeneous cores are one possible solution to provide both low latency and high throughput. Some number of aggressive cores provide low single-thread latency, while an array of small, multi-threaded in-order cores provide throughput. This configuration, however, limits the number of

concurrent performance-sensitive threads to the number of aggressive cores, limiting the number of applications that are a good fit.

In the presence of single-thread latency constraints, this work shows that it is actually aggressive, out-of-order cores that provide not only the necessary performance, but also the best, or at least competitive, aggregate throughput. This is true even when the latency constraint is quite generous, e.g. a 10X slowdown. It is only when latency constraints are negligible and throughput is paramount that in-order cores are superior.

There are three major reasons that contribute to the superiority of OO designs:

- Higher IPC
- BIPS/power/area efficiency
- Lower relative interconnect-fabric overhead

This work demonstrates the superiority of a symmetric, OO multi-core solution for performance-sensitive markets and performs a sensitivity study to verify this conclusion. This holds true for both CPU-bound and memory-bound applications unless an in-order core is at least more than 25% smaller than an OO core. This work also shows that the penalty for selecting a compromise configuration for best overall performance (across different workloads and different latency/throughput requirements) is acceptable and dynamically turning on and off the SMT mechanism for a CMP based on out-of-order SMT cores may provide the best balance between single thread performance and chip throughput.

To reach these conclusions, IBM's Turandot/PowerTimer model and University of Virginia's multi-core simulation extensions are used in this work. Turandot/PowerTimer has been scaled to model in-order cores and matched against various published data. Results are obtained from an exhaustive search of the large multi-core design space consisting of number of cores, pipeline depth, superscalar issue width, in-order vs. out-of-order issue, non-SMT vs. SMT, size of L1 and L2 caches. This work optimizes for throughput subject to single-thread latency, peak steady-state temperature, power delivery, pin-bandwidth, and area constraints.

The rest of this chapter is organized as follows. Section 7.2 introduces the related work. Section 7.3 outlines experiments methodology including a detailed discussion of performance, area, and power scaling. Section 7.4 discusses results and includes a detailed sensitivity analysis. Finally this work is concluded in Section 7.5.

7.2 Related Work

There has been a burst of work in recent years to understand the performance, energy, and thermal efficiency of different CMP organizations. Few have looked at a large numbers of cores.

Davis et al. [15] explore the design space for core type, number of cores, cache size, and degree of multithreading, but focus on maximizing throughput without regard for single-thread latency. They show that simple, shallow, in-order cores with large numbers of threads per core are optimal. The reason is that multiple threads allow servicing of multiple cache misses to be overlapped with modest hardware: only an extra register set per thread, instead of the expensive out-of-order hardware required to exploit substantial memory-level parallelism within a single thread. This work focused on transaction processing (OLTP) workloads, which tend to have poor instruction-level parallelism and poor cache locality, and found 4–8 threads per core to be optimal depending on workload. Several existing products embody this philosophy. Kongetira et al. [38] describe the Sun T2000 “Niagara” processor, an eight-way multi-core chip supporting four threads per core and targeted toward workloads with high degrees of thread-level parallelism. Chaudhry et al. [13] go on to describe the benefits of both multiple cores and multiple threads and sharing eight cores with a single L2 cache. They also describe the Sun Rock processor’s “scouting” mechanism that uses a helper thread to prefetch instructions and data [13]. Graphics processors (GPUs) also embody this philosophy, with large numbers of fairly general-purpose “shaders” (i.e., cores) and the ability to keep many threads in flight. For example, the ATI R580 exhibits 56 shaders and can support 512 concurrent threads (where each thread is servicing a pixel), while the Nvidia G71 series exhibits 32 shaders (but a larger number of texture units). Like the T2000, GPUs stress throughput over single-thread (single-pixel) latency, and use the high degree of multithreading to mask memory (chiefly

texture) latency.

We [53] also explore the design space for core count, pipeline depth, out-of-order issue width, and L2 size, and show the importance of thermal constraints, but only consider single-threaded cores. Our work focuses on single-threaded, multi-programmed SPEC workloads. Monchiero et al. [57] explore a similar design space and also demonstrate the importance of thermal constraints, but this time in the context of an assortment of parallel shared-memory applications. Li and Martínez [49] instead focus on power constraints, but study the SPLASH parallel benchmarks. Their results show that parallel execution on a CMP can improve energy efficiency compared to the same performance achieved via single-threaded execution, and that even within the power budget of a single core, a CMP allows substantial speedups compared to single-threaded execution. In [50], they go on to develop heuristics for dynamic adaptation to allow a CMP to find the optimal voltage/frequency settings and optimal number of cores to put to sleep to meet performance constraints while maximizing power savings. In a related work, Donald and Martonosi [18] develop heuristics for scheduling threads on a CMP to minimize thermal throttling, while Powell et al. [62] instead propose a core-hopping approach on CMPs in response to imminent thermal throttling.

Huh et al. [32] categorized the SPEC benchmarks into CPU-bound, cache-sensitive, or bandwidth-limited groups and explored core complexity, area efficiency, and pin-bandwidth limitations, concluding, as this work does, that out-of-order cores are generally preferable because of their greater area efficiency. Ekman and Stenstrom [20] use SPLASH benchmarks to explore a similar design space in the context of energy-efficiency, arriving at the same conclusions. These papers did not, however, account for the area overhead of on-chip memory controllers, since that is a fairly recent phenomenon.

The methodologies for analyzing pipeline depth and width build on prior work by Lee and Brooks [47] by developing first-order models for capturing changes in core area as pipeline dimensions change, thereby enabling power density and temperature analysis. This work identifies optimal pipeline dimensions in the context of CMP architectures whereas most prior pipeline analyses consider single-core microprocessors [25, 28, 72]. Furthermore, most prior work in optimizing pipelines focused exclusively on performance although Zyuban et al. found 18FO4 delays to be

power-performance optimal for a single-threaded microprocessor [77].

7.3 Methodology

In order to allow tractable simulation of such a large number of cores and core designs, this work employs a simulation methodology that allows decoupling the simulation of individual cores with the chip fabric and L2 simulation [53]. This methodology uses IBM's Turandot/PowerTimer simulator, a detailed, cycle-accurate, execution-driven simulator to collect the IPC and power statistics in chunks of 10k instructions. The simulator also generates single-core L2 cache access traces. Zauber, the shared L2 and fabric simulator, uses this data to collect performance and power results for various CMP configurations based on these single-core L2 cache access traces and statistics. By separately simulating the L2 and the fabric, Zauber can be used to accelerate CMP simulation. In this work a simplified temperature model [53] validated against Hotspot 2.0 is used to estimate the temperature at the core granularity. Turandot and PowerTimer was originally designed to model a POWER4 like out-of-order architecture, and for this research, the toolset is modified to simulate in-order architectures.

7.3.1 Different Architectures

This study models architectures listed in Table 7.1. As this table shows, this work considers the following core design choices: SMT or non-SMT, in-order or out-of-order, issue widths (2 way or 4 way), and different Dcache sizes (8KB or 32KB). Pipeline depth is also varied from 18FO4 to 42FO4 in a step of 6FO4 for each design. This study only considers 2-way SMT. If the issue width is 4, each thread will occupy two pipelines on average and if the issue width is 2, each thread will occupy a single pipeline. In this way, the thread number per pipeline can be changed and this study will show the impact of this factor on each architecture. Resource sizes and power are scaled from non-SMT to SMT following the same methodology we proposed in [52]. The area and power scaling methodology in this work is the same as our methodology in [53] when the pipeline depth and width of each design are changed.

	OO4	OO2	OS4	OS2	OO4SMT	IO4	IO2	IS4	IS2	IO4SMT
Architecture	OO	OO	OO	OO	OO	IO	IO	IO	IO	IO
Dcache	32KB	32KB	8KB	8KB	32KB	32KB	32KB	8KB	8KB	32KB
Issue Width	4	2	4	2	4	4	2	4	2	4

Table 7.1: **Terminology**

7.3.2 In-order Architecture Modeling

7.3.2.1 Performance Model for an In-order Architecture

The main difference between an IO architecture and an OO architecture is that the IO architecture blocks issue if a previous instruction is stalled because of instruction dependency. The issue logic of the Turandot model is changed to enforce in-order execution, which means if there is an instruction waiting for operands, all other instructions after this instruction in the program order will be blocked; while in the case of out-of-order execution, these instructions do not need to wait for the issue of that previous instruction. The in-order model in this work still allows out-of-order retirement. Instructions in modern processors tend to have quite different execution latencies and out-of-order retirement can improve the performance of an in-order architecture with very little hardware overhead. Besides these changes to the fundamental pipeline logic, sizes of some resources are also scaled down compared with OO architectures because unlike an OO core, an IO core does not require as many queue resources. For the same issue-width, this work scales down the physical register file size, load/store queue size, data miss queue size and retirement queue size by half from OO to IO. The issue queue size is set to the issue width because a larger issue queue can barely help IO architectures due to the intrinsic property of in-order issue.

7.3.2.2 Power and Area Model for an In-order Architecture

We need to scale down the unconstrained power (e.g. power before clock gating effects are considered) for all resources whose sizes are changed from the baseline OO model. In this work CACTI is used to calculate the scaling factor for the L1 data cache (Dcache), L1 instruction cache (Icache), first-level data TLB (DTLB), and first-level instruction TLB (ITLB). For resources like functional units, this work assumes linear scaling according to the size or the number of structures. However,

	OO4	OO2	IO4	IO2	OO4SMT	OO2SMT	IO4SMT	IO2SMT
32KB	11.22	8.38	9.85	7.46	12.89	9.33	10.61	8.26
8KB	8.68	6.45	7.31	5.52	10.35	7.39	8.07	6.32

Table 7.2: **Core area of different architectures at 19FO4** (mm^2)

linear power scaling is inaccurate for certain non-clustered structures. For example, it does not capture non-linear relationships between power and the number of SRAM access ports since it does not account for the additional circuitry required in a multi-ported SRAM cell. For this reason, this work applies superlinear power scaling with exponents drawn from Zyuban's work in estimating energy growth parameters [79]. Since these parameters were experimentally derived through analysis of a non-clustered architecture, this power scaling is only applied to the non-clustered components of the architecture. This methodology does not consider the power benefits of decreasing hardware complexity for certain structures from OO to IO. For example, issue queues in an IO architecture can be much simpler due to simpler wakeup and issue logic. However, because the sizes of those resources in the IO model are very small (2-4 in the case of issue queues), further scaling of their power changes the total power consumption estimation very little.

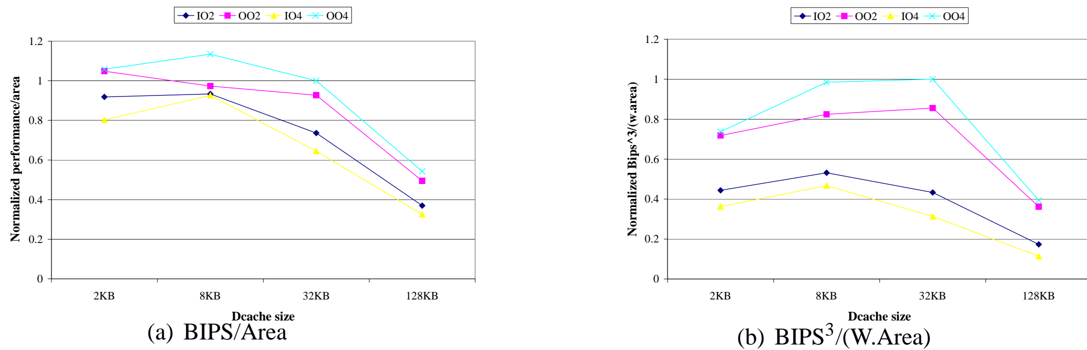
This work follows a similar methodology to scale down the core area from the OO model to the IO model. Sensitivity analysis for area and power models of IO architectures is presented in Section 7.4. Finally, all experiments are performed in the 65nm technology node. Table 7.2 shows the area for cores with different architectures in this technology.

The core area may only occupy less than 50% of the chip area in CMP designs. It is very important to model the area of other on-chip structures. This research models the area for on-chip L2 cache, DDR and DRAM controllers, and on-chip interconnects. The on-chip L2 cache area is estimated based on the POWER4 die photo and scaled to 65nm. Because this work assumes every 4 cores share an L2 cache, 4-way crossbars are used to accomplish on-chip interconnection. It is assumed that these 4-way crossbars will be implemented in a higher metal layer over the L2 cache as in Kumar et al. [45]. The area estimation for DDR/DRAM controllers is based on the die photo of the Sun Niagara and its total chip memory bandwidth. A linear area scaling of DDR/DRAM controllers is assumed relative to the chip's maximum pin-bandwidth.

1MB L2 cache	DDR/DRAM controller for 1GB/s Pin Bandwidth	4-way Crossbar
11.52	0.96	22.80

Table 7.3: **Area of non-core structures on a CMP chip** (mm^2)

The level one instruction and data cache structures can be responsible for more than half the core area and therefore properly selecting sizes for these structures is critical to the chip area efficiency. An experiment is carried out to sweep several cache sizes to see how the performance and BIPS^3/W change with L1 cache size. The sizes of all on-core cache structures are changed by the same factor in this experiment. As shown in Figure 7.1, the best BIPS/Area and BIPS^3/(W.Area) are achieved with 8KB or 32KB L1 data caches for almost all architectures. These BIPS/Area and BIPS^3/(W.Area) values are normalized against the case of OO4 with 32KB data cache. Therefore all following experiments in this work assume these two L1 cache sizes.



(a) BIPS/Area

(b) BIPS^3/(W.Area)

Figure 7.1: **Normalized BIPS/Area and BIPS^3/(W.Area) with different cache size**

7.3.3 Physical Constraints

This research considers several physical constraints: chip temperature, chip power, chip area, and off-chip pin-bandwidth. The total chip power limit is set at 250w and the chip peak temperature is set at 100C. If these peak values are reached voltage scaling will be used to throttle the chip speed so these limitation can be maintained. This work tries two different chip area constraints : 400mm^2 and 200mm^2, corresponding to two different chip markets for CMP chips. Three different pin-bandwidth limitations are tried in this study: 24GB/s, 48GB/s, and 96GB/s. Different pin-

bandwidth corresponds to different chip area used by on-chip DDR/DRAM controllers as indicated by Table 7.3. If the pin-bandwidth limitation is reached the core frequency will be throttled back to ensure that the average pin bandwidth requirement does not exceed the maximum chip pin-bandwidth. This work does not consider the burdens of bursty traffic on chip pin-bandwidth.

7.3.4 Benchmark Methodology

Table 7.4 lists four benchmarks this study uses to represent four typical CMP application scenarios. They are SpecJBB and mcf, art, and gcc from Spec2K. To generate these traces, these benchmarks are compiled with the *xlc* compiler and -O3 option. Simpoint [23] is used to identify representative simulation points and traces are generated by capturing 100 million instructions beginning at the Simpoint for all Spec2k benchmarks. The SpecJBB benchmark used in this research does not have phase change behavior for the whole trace and therefore this study simply picks a segment with 100 million instructions by skipping a large number of instructions from the beginning of the trace. Each benchmark is replicated to form CMP benchmark tuples.

As mentioned earlier, single-thread latency may matter for more than just legacy single-threaded applications. This work selected single-threaded applications that exhibit task-level parallelism but still require low single-thread latency. After characterizing the design space for these four applications, this work found SpecJBB and mcf to best represent the range of behavior that was observed.

- SpecJBB represents an e-commerce workload. As mentioned earlier, some markets and customers will require fast response times. Measurements suggest that 25-50% of response time for Internet services can be CPU time. We chose SpecJBB as a focus because it is insensitive to L1 cache size and exhibits a significant rate of off-chip traffic regardless of L2 cache size, while still benefiting from the ILP benefit of OO cores.

- mcf computes a minimum-cost-network for traffic routing and hence is representative of decision support applications that may require interactive response, or continuously refine their answer indefinitely, until the best available answer is required. We chose mcf as a focus be-

Benchmark name	Description
SpecJBB	E-commerce server
mcf	Decision support
art	Data analysis and data mining: image processing
gcc	Interactive server

Table 7.4: **Description of benchmarks**

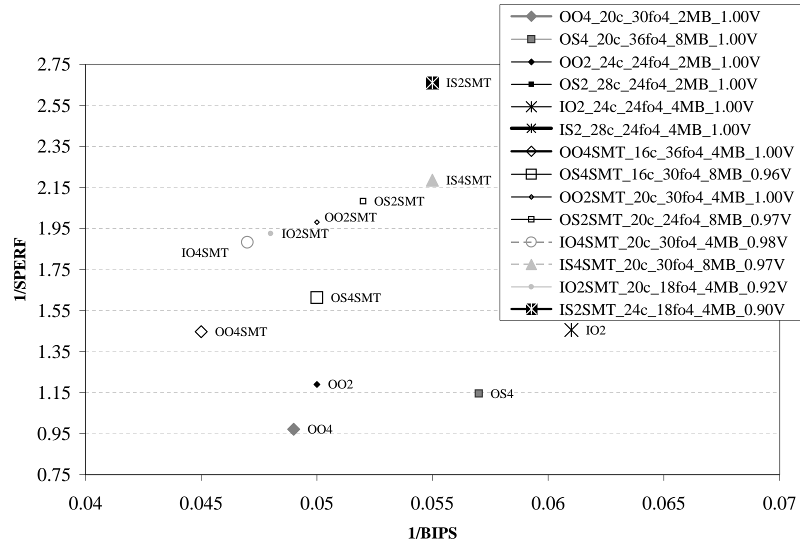
cause it has a 2MB working set, and hence is extremely memory bound unless each thread can have 2 MB of L2 cache.

- gcc is representative of applications that do not parallelize easily. Single-thread latency matters for large files that determine response time. (sim-outorder.c is one example many readers are likely to be familiar with)
- art performs image recognition and is representative of applications that may parallelize well, but can run equally well as single-threaded applications when a large number of independent tasks are present. Single-thread latency then matters for interactive response.

7.4 Results

This section presents detailed results for the design space study under the modeled constraints. It starts by considering the single-thread latency and maximum throughput for each of the base architectures and then considers the sensitivity to all key design assumptions and constraints; specifically, this study considers the impact of thermal constraints, pin-bandwidth, area assumptions, and the impact of disabling SMT.

The amount of data required to exhaustively explore all of these constraints prohibits us from presenting all results. This work has studied the data carefully to identify representative trends. The data for two of key benchmarks are presented: SpecJBB and mcf. Section 7.4.5 considers the selection of an optimal architecture that includes all four benchmarks.



(a) Latency vs. Throughput for SpecJBB with 400sqmm die size and an expensive low-resistance (LR) heatsink.

Figure 7.2: **Single thread latency (y-axis) vs. Total throughput (x-axis). Best performance towards the origin.**

7.4.1 Understanding the design space

Figure 7.2 plots the single-thread latency and total throughput for SpecJBB for all base architectures. Each point in this figure displays the throughput-optimized configuration after applying thermal design constraints. The legend in the figure displays the exact configuration in terms of the number of cores, the pipeline depth of each core, the L2 cache size, and the amount of DVS throttling required to meet thermal constraints.

From this figure, we see that the out-of-order configurations tend to have both the best single-thread latency and the highest throughput. This is partly because the out-of-order cores are designed with relatively shallow pipelines and modest L2 caches. But the main reason here is the inherent better performance and $BIPS^3/W$ per area of the OO architecture for the benchmarks that are investigated. For almost all benchmarks, OO is an area efficient way to improve IPC.

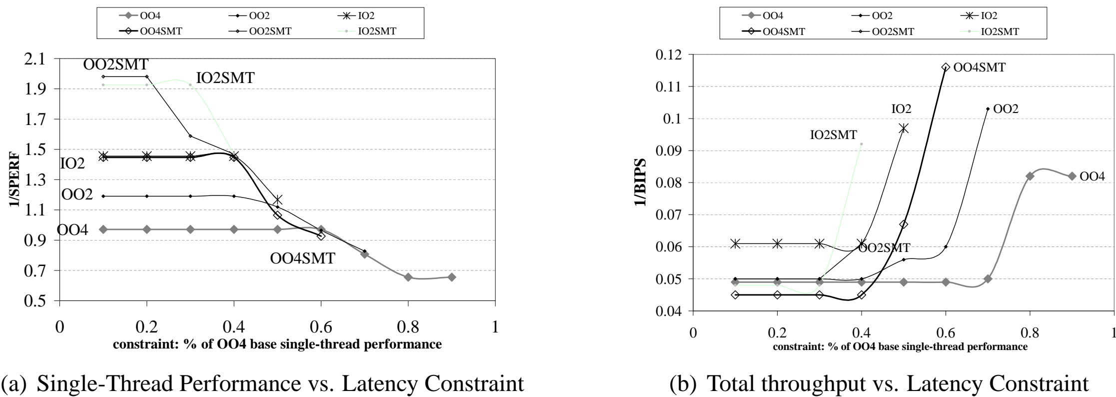
Introducing SMT helps improve throughput for both OO and IO. Keeping the multithreading degree at 2 and increasing the issue width from 2 to 4 will generally help improve throughput for OO. However, we see that increasing the thread number per pipeline does not necessarily help IO

cores (IS2SMT and IS4SMT) because IO cores do not favor wider pipelines to exploit instruction level parallelism in general.

7.4.2 Sensitivity to latency constraints

This section considers the sensitivity of the results to designs where single-thread latency restrictions are placed on the optimization procedure. Specifically, this study places the restriction that a design must have single-thread performance within *n%* of a previous generation design; in this case, this study chooses the POWER4-like baseline as this design and sweeps *n* from 10% to 90%.

For each of these designs, this work optimizes for total throughput after meeting the latency constraint. Note that for some values of *n*, some of the architectures may not be able to meet the latency constraint and their results are not shown. After applying this constraint, two metrics of performance are considered; the resulting single-thread latency and the total throughput.

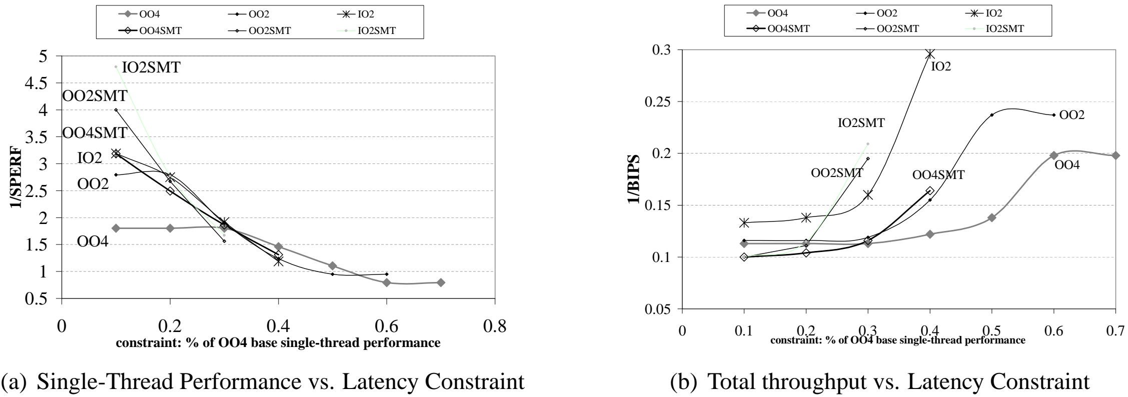


(a) Single-Thread Performance vs. Latency Constraint

(b) Total throughput vs. Latency Constraint

Figure 7.3: **SpecJBB with 400sqmm die and LR heatsink**

Here I first present results for SpecJBB with both the low-resistance (LR) (Figure 7.3) and high-resistance (HR) (Figure 7.4) heatsinks. Figure 7.3a shows the single-thread latency for each architectural configuration. We see that with *n* larger than 50% no in-order configurations are viable. We also see the well-known trend that SMT architectures can hurt single-thread latency: the best in-order SMT configurations can only meet the 40% latency constraint and even the OO4 SMT configuration is only able to meet the 60% latency constraint.



(a) Single-Thread Performance vs. Latency Constraint

(b) Total throughput vs. Latency Constraint

Figure 7.4: **SpecJBB with 400sqmm die and HR heatsink**

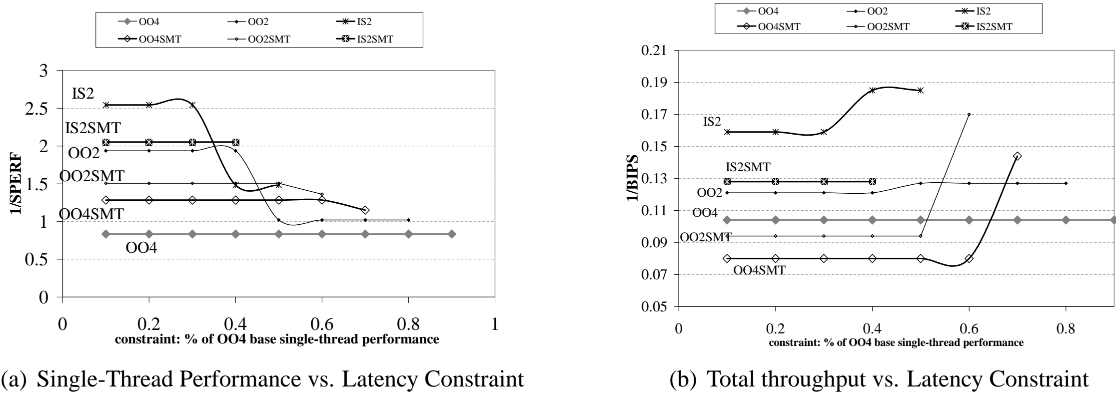
Figure 7.3b shows the optimized throughput for each of these identical configurations. We see that the OO4 SMT configuration achieves the best throughput up to the 40% constraint; after that, the simple OO4 configuration achieves better throughput. The best in-order configuration, IO2 SMT, is competitive with each of the other out-of-order architectures, but lags OO4-SMT by about 10% in total throughput. We also see that in-order configurations without SMT are substantially inferior. If the design requires that the single-thread latency be closer to the previous generation (e.g. within 50%), then the OO4 configuration achieves the best throughput. The reason for this is that OO4 provides the best single thread performance if there is no throughput requirement, therefore, if strict single thread requirement is enforced, other architectures must sacrifice throughput for single thread performance (for example, by moving to a deeper pipeline or upsizing L1 caches) while the OO architecture can still maintain its near peak throughput.

Figure 7.4a shows the same scenario except that we have imposed harsher thermal constraints by replacing the low-resistance heatsink with a high-resistance heatsink. Compared to the LR case, many more configurations are eliminated because many suffer severe thermal throttling causing DVFS to be engaged – even though we compare to a baseline machine that also has an HR heatsink, the number of cores in the throughput-optimized designs causes enough global heatup within the heat spreader to cause additional throttling.

Overall, we see that the more thermally-constrained design somewhat levels the field between the in-order and out-of-order designs for single-thread latency; in fact, the IO2 design achieves the

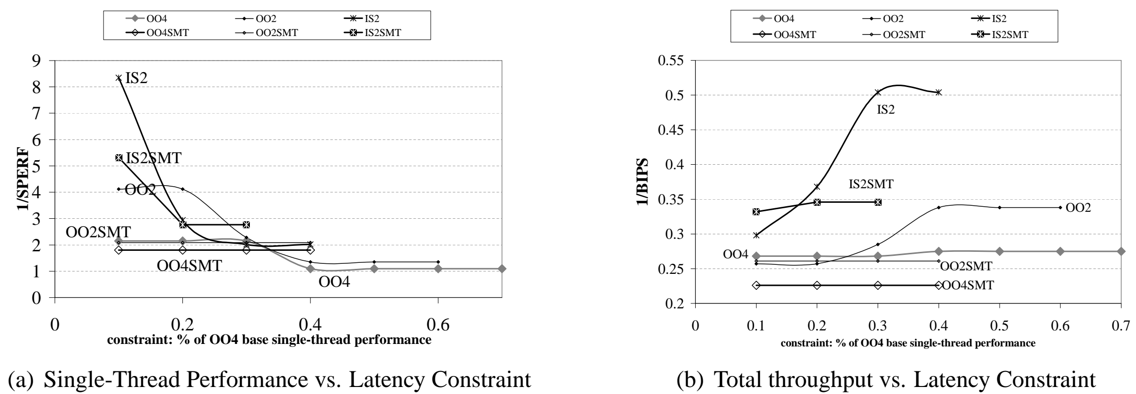
best single-thread performance at the 40% point and the IO2 SMT design is very close to the best at the 30% point. At the 50% point, all the in-order designs are eliminated, but the OO2 design is better due to its superior power characteristics. Thus, under severe thermal constraints, simpler cores can beat out the OO4 design for single-thread performance.

Figure 7.4b shows the total throughput with the HR configuration. We can find that the IO2 SMT and the OO4 SMT configuration are comparable with the 10% latency constraint. With latency constraints less than 30%, many configurations are quite close and OO4 is only clearly better when the latency constraint is 40% or higher.



(a) Single-Thread Performance vs. Latency Constraint (b) Total throughput vs. Latency Constraint

Figure 7.5: **MCF with 400sqmm die and LR heatsink**



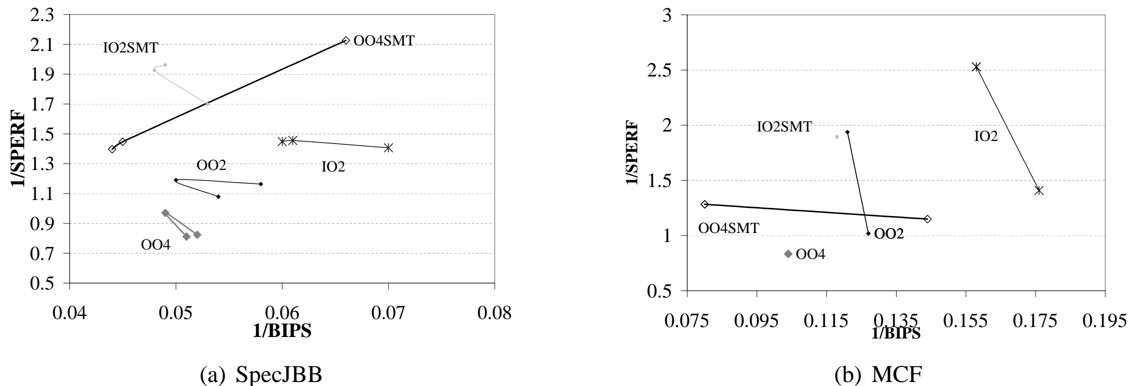
(a) Single-Thread Performance vs. Latency Constraint (b) Total throughput vs. Latency Constraint

Figure 7.6: **MCF with 400sqmm die and HR heatsink**

Figures 7.5 and 7.6 present similar results for the mcf workload. Overall, many of the same trends that we observe for SpecJBB hold; there is a wide spread in single-thread performance

between in-order configurations and out-of-order. When considering total throughput, the OO SMT configurations are clearly the best choices even with the 10% latency constraint. With the HR heatsink configuration, we can find that because of the overall decrease in performance and IO's power efficiency, IO architectures are more competitive relative to OO, but in no case do they surpass the OO architectures for throughput. Mcf is a memory bound benchmark and tends to choose a big L2 cache as the optimal configuration. This further mitigates the area advantage of IO architectures because in this case L2 cache occupies a big portion of the chip area and that leads to big throughput difference between IO architectures and OO architectures.

7.4.3 Sensitivity to bandwidth constraints



(a) SpecJBB (b) MCF

Figure 7.7: **Pin-Bandwidth Constraints with 400sqmm die and LR heatsink. Each point represents one of 24, 48, or 96GB/s total chip bandwidth. The 48GB/s point is always the middle point in groups of three.**

Pin-bandwidth limitations are likely to be an increasing challenge for designers of multi-core CPUs. This section considers the sensitivity of results to pin-bandwidth constraints. Increased pin-bandwidth is modeled by increasing the total number of DDR channels on the die; these additional channels cost additional area that may restrict the number of cores or L2 cache on the chip. Thus, more pin-bandwidth can actually be detrimental to total throughput in some cases.

Figure 7.7 shows the results of this analysis for SpecJBB and mcf. We see that in some cases, more pin-bandwidth is absolutely essential; for example, OO4 with SMT requires 96GB/s to achieve maximum potential for SpecJBB. In other cases, the additional area overhead of the DDR

channels is not worthwhile; for example, IO2 SMT and OO2 achieve better throughput with 48 GB/s than either 24GB/s or 96GB/s. In general, we see that pin-bandwidth is less of an issue for mcf, and many designs achieve better throughput with less bandwidth/DDR controllers. The optimal configurations for mcf usually provides enough L2 cache size to contain its working set. We can find that although it is an L2 cache bound benchmark, as long as the working set is held in the L2 cache, off-chip pin-bandwidth requirements are quite low.

7.4.4 Sensitivity to in-order core size

In-order cores tend to be 30% to 50% smaller than OO cores, depending on the L1 cache size that is used. Because of this, reducing the area of other non-core on-chip structures will help improve the relative area advantage of in-order architectures. Figures 7.8 and 7.9 investigate the performance sensitivity to the area of IO cores and other on-chip non-core structures. First, the interconnection area and power are reduced to 10% of the default assumption. The L2 leakage power is also reduced to 10% of the default assumption. Then this study sweeps three different IO core sizes: 50%, 70%, and 90% of the original IO core sizes. Each case is represented by one point on the same line for all IO architectures. But the area of the OO cores is not changed. Pin-bandwidth is set to be 48GB/s and total chip area is set at $200mm^2$ for this experiment.

As we can see from these figures, reducing the area of non-core structures gives IO architectures more area advantage. Figure 7.8a shows that even for the case that IO core size is only scaled to 90% of the original size, the optimal throughput of IO2SMT is still more than 20% better than the best OO configurations. This trend also holds in the HR case as shown by Figure 7.8b. Reducing the area of IO cores improves the performance of IO architectures even more. Figure 7.8 shows that the performance of IO2SMT and IO2 can improve by 30% to 90% when we scale the IO area from 90% to 50%. The throughput of IO2SMT with the most optimistic IO area estimation can beat the best OO's throughput by 100% as shown in Figure 7.8b. However, in almost all cases, the optimal IO configurations for throughput always have worse single-thread performance than OO architectures.

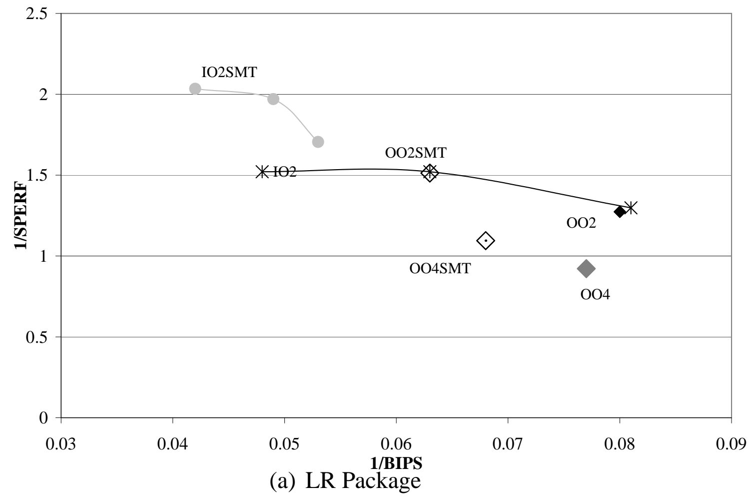
But if we look into Figure 7.9, we will find that even if we assume IO cores are only 50% of

their original size, the throughput of the best IO configuration is still slightly worse than the best OO configuration. As mentioned above, mcf is a memory bound benchmark and tends to choose a big L2 cache as the optimal configuration. Therefore the area benefits of scaling down IO core size for mcf is not as big as for JBB. The most interesting example to show this effect is IS2SMT in Figure 7.9a. Here the throughput of IS2SMT does not change at all even its core area is scaled from 90% to 50%. Adding more cores while keeping L2 cache size unchanged leads to much higher cache miss ratio, negative total throughput return and higher pin-bandwidth requirements while adding more L2 cache to hold working sets from additional cores will exceed the chip area constraint. More severe thermal constraint can favor IO architectures as shown in Figure 7.9b. But even in this case, only IS2 with the most optimistic area estimation can win OO by less than 10%.

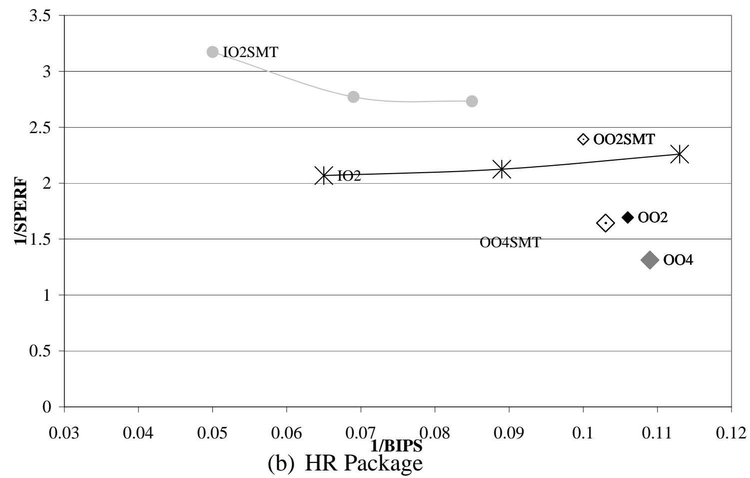
7.4.5 Optimal tradeoff configurations

This section finds the optimal tradeoff configurations for all benchmarks. Table 7.5 lists the best configurations for each architectures with different thermal packages. In this experiment, pin-bandwidth limits are set at 48GB/s and chip area is set at $400mm^2$. If a configuration achieves the best average throughput for all four benchmarks it the best configuration across all benchmarks. From this table, we see that the optimal configurations require a core count from 16 to 20 with a moderate L2 cache size at 8MB. However, there are two outliers, IO2+HR and IS2+HR, both of which require many cores and a very small L2 cache size. We also see that a cheap thermal package (HR) usually requires a shallower pipeline (24FO4-36FO4) while an expensive thermal package needs a deeper pipeline (18FO4-24FO4) because shallow pipelines have power and therefore thermal advantages. This study also compares the performance achieved by these optimal tradeoff configurations and those optimal configurations for each specific benchmark. This result is shown in Figure 7.10. This figure shows that the difference is negligible for the HR case (with around 0.5% loss) and moderate for LR cases (around 1-4% loss). LR always leads to more loss for the same architecture because fewer thermal constraints give more possible configurations and that leads to more performance diversity in the whole design space.

While previous results indicate that OO/IO with SMT support is a very efficient way to achieve

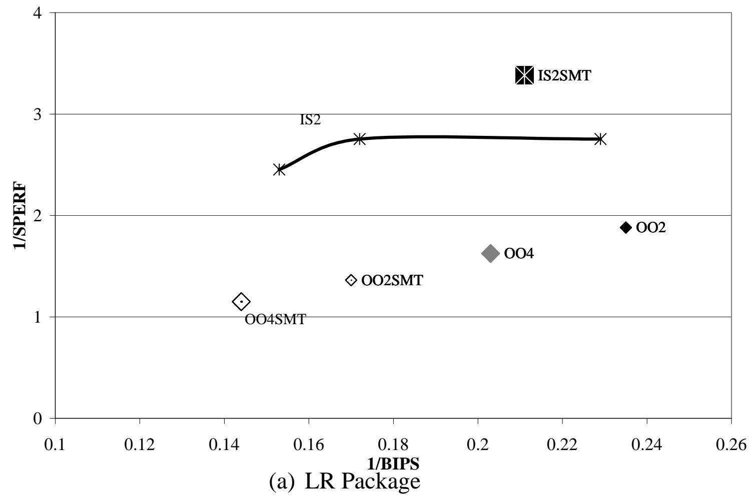


(a) LR Package

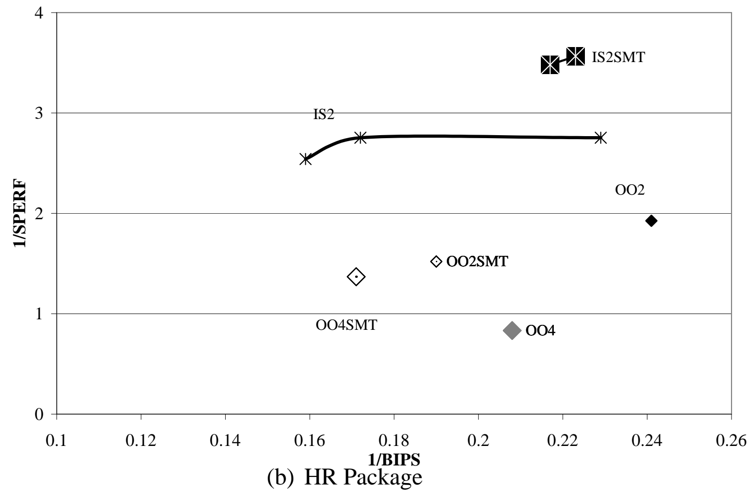


(b) HR Package

Figure 7.8: **Area sensitivity (specJBB). Each point on the same line represents scaling IO core area down to 50%, 70%, and 90%.**



(a) LR Package



(b) HR Package

Figure 7.9: **Area sensitivity (MCF). Each point on the same line represents scaling IO core area down to 50%, 70%, and 90%.**

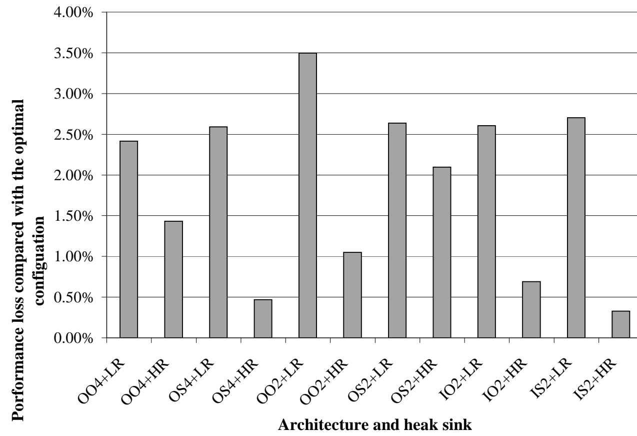


Figure 7.10: **The performance loss using optimal tradeoff configurations**

	Core count	L2 cache	FO4
OO4+LR	16	8	24
OO4+HR	16	8	36
OS4+LR	16	8	24
OS4+HR	20	8	36
OO2+LR	16	8	18
OO2+HR	20	8	36
OS2+LR	20	8	18
OS2+HR	20	8	36
IO2+LR	16	8	18
IO2+HR	24	2	24
IS2+LR	24	8	24
IS2+HR	28	2	24

Table 7.5: **Configurations to achieve the best average performance across benchmark**

good throughput, we know that in certain circumstances we want to maintain good single thread performance. Therefore, users would like to be able to turn off SMT dynamically. The performance impact of turning off SMT for an SMT based CMP chip is shown in Figure 7.11. In this figure, each line contains two points and the point with better single thread performance and worse throughput is always when SMT is turned off and vice versa. As we can see from this figure, for all architectures, turning off SMT will lead to improved single thread performance, usually by 20% to 50%, but that will also trigger a throughput loss of 10% or so. This clearly shows that dynamically turning off SMT is a useful technique to maintain both good throughput and single thread performance.

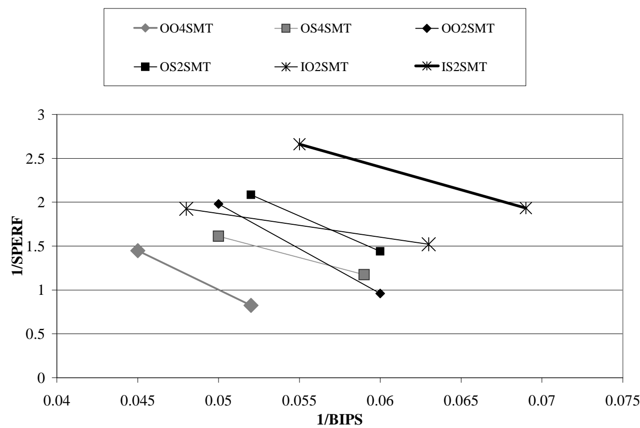


Figure 7.11: **Turning off one thread in SMT (points lower on the y-axis are with SMT disabled)**

7.5 Future Work and Conclusions

This work provides a comprehensive analysis of CMP design when considering performance in terms of both single-thread latency and aggregate chip throughput. This analysis is performed under a variety of technological constraints, including area, thermal, energy, and pin-bandwidth limitations. Considering such a large space of parameters requires significant infrastructure development and careful attention to experimental methodology.

Overall, this study finds that in many cases, conventional out-of-order architectures are superior for both throughput and latency when considering all constraints. This study finds that the OO cores tend to be relatively area and power efficient compared to IO cores and that SMT can provide significant benefits to both core styles. Thermal constraints can have a significant impact on the design space study, and in some cases, can allow in-order cores to give equivalent throughput to the best OO core. These results suggest that while in-order cores may be appropriate for purely throughput-oriented applications, market segments that require a mixture of single-thread latency and throughput may be best served by traditional out-of-order core designs.

Chapter 8

Conclusions and Future Direction

8.1 Dissertation Summary

In summary, this dissertation reaches the following conclusions:

- Joint optimization across multiple design variables is necessary. Even pipeline depth, typically fixed in architecture studies, may impact core area and power enough to change the optimal core count. Optimizing without thermal constraints and then scaling to a thermal envelope leads to dramatically inferior designs compared to those obtained from including thermal constraints in the initial optimization.

- Thermal constraints tend to favor shallower pipelines and narrower cores, and tend to reduce the optimal number of cores and L2 cache size. Nevertheless, even under severe thermal constraints, additional cores benefit throughput despite aggressive reductions in operating voltage and frequency. This is true until performance gains from an additional core is negated by the impact of the additional voltage and frequency scaling required of all the cores. This inflection occurs at approximately 55% of the nominal Vdd, well into the range of non-linear frequency scaling (18% of nominal!).

- For aggressive cooling solutions, reducing power density is at least as important as reducing total power. For low-cost cooling solutions, however, reducing total power is more impor-

tant because raising power dissipation (even if power density is the same) raises a chip's temperature.

- In many cases conventional out-of-order architectures are superior in terms of both throughput and latency when considering all constraints. OO cores tend to be relatively area and power efficient compared to IO cores for a wide range of benchmarks.

- Thermal constraints can have a significant impact on the design space study, and in some cases, can allow in-order cores to give equivalent throughput to the best OO core.

- SMT can provide significant benefits to both core styles. For the OO architecture, SMT can provide a performance speedup of nearly 20% for a wide range of applications with a power overhead of roughly 24%. Thus, SMT can provide a substantial benefit for energy-efficiency metrics such as ED^2. The performance-energy optimal resource scaling factor was found to be at 1.5 for SMT. Dynamically turning on and off the SMT mechanism for an SMT-core based CMP is a promising technique to achieve both high throughput and good single thread performance.

- CPU- and memory-bound applications have quite different optimal configurations. Adaptivity helps, but any compromise incurs throughput penalties. However, with thermal limits, the optimal configurations begin to converge.

- The mechanisms by which SMT and CMP chips heat up are quite different. More specifically, SMT heating is primarily caused by localized heating in certain key structures, while CMP heating is mainly caused by the global impact of increased energy output. Because of this difference in heat up mechanisms, this dissertation finds that the best thermal management technique is likewise different for SMT and CMP. Indeed, non-DVS localized thermal-management can outperform DVS for SMT.

8.2 Future Work

8.2.1 Methods to accelerate CMP design space searching

This dissertation proposes a CMP simulation acceleration technique called Zauber. The current Zauber can only simulate CMP with independent threads. However, CMP is also an infrastructure to run parallel applications. There have been several works trying to investigate the performance and power efficiency of CMP for parallel applications. Due to infrastructure limitation, most research has only looked into CMP with less than 8 cores and few research has looked into cases more than 16 cores. Enhancing Zauber to accommodate parallel applications is a challenging future task.

The simulation models described in this dissertation are all event-driven. Even for a simplified simulator that only models events in the cache hierarchy, simulating realistic behavior in large caches requires long traces. To search very large design spaces with the fabric simulator is therefore costly, possibly prohibitively so. There has been some research work trying to search the CMP design space using traditional optimization techniques like hill-climbing [43]. Another possible way is to develop an analytical model that estimates performance, energy, and thermal metrics with a simple equation rather than costly event-driven modeling. Deriving this analytical model may still require a bounded number of fabric simulations for each workload of interest, but this is far less costly than simulating each design point with the fabric simulation.

Several possible approaches exist for developing an analytical model. All of them entail sampling the design space with fabric simulations in order to obtain accurate estimates of the metrics for representative configurations. Selecting points which span the design space in a representative way, and determining how many points are needed, will be guided by sampling theory or other statistical techniques. One simple approach for deriving an analytical model is to develop an equation that describes how changes in design configuration will change metrics of interest. This is essentially a sophisticated form of classic performance equations like those taught in computer architecture courses. For example, reducing cache size will reduce power dissipation (but not power density) due to the smaller structure, but it will also increase capacity misses, and increase contention events throughout the hierarchy. These will introduce stalls, which in turn increase energy consumption.

The impact of the stalls and increased energy must be scaled according to different workloads' latency tolerance and different core styles and their clock gating. Using the metrics' values at representative design points allows derivation of the coefficients for each term. Developing this form of analytical model therefore identifies for designers the different factors that contribute to each metric and their relationship. A challenge with this approach is that the coefficient values may vary across the design space. This could require the coefficients themselves to be functions of the design parameters, or require some kind of interpolation.

An alternative approach is to omit an analytic performance equation as described above, and simply interpolate between the metrics' values at representative design points. This avoids the problem that the coefficients in a performance equation may vary in ways that are difficult to express analytically. Instead, this approach depends on finding representative fabric-simulation points, between which the coefficients variation can easily be captured by an interpolation function. A drawback to this approach is that it does not give designers an equation that describes how the metrics depend on the design parameters. A third approach combines the previous two. Instead of directly predicting the metrics using a performance equation, we can predict how the metrics change. But unlike interpolation, an expression is derived to predict this change based on how the design changes. This provides an equation that describes how the metrics depend on the design parameters, but makes better use of the ability to find representative design points.

The final step in using an analytical model is to identify the most interesting design regions and zoom back in to evaluate them with increasing fidelity, until finally simulating in full detail a small number of leading candidates.

8.2.2 More design dimensions

Most important CMP design dimensions are investigated in this dissertation, but there are several dimensions left out. First, heterogeneous CMP is a possible solution to accommodate different benchmarks. Adding heterogeneous CMP as a possible CMP design dimension greatly expands this design space. Questions related to how many different types of cores are needed and how to combine and organize different cores will need to be answered. It will be impractical to solve

these problems until we have a better infrastructure for exploration, such as the one proposed in the previous section.

Second, run-ahead is a promising technique to improve per-thread performance using CMP infrastructures. what kind of run-ahead techniques are the best ones for CMP. When should we launch a run-ahead thread in terms of performance-energy efficiency. Should we limit the run-ahead technique within one core or can we distribute run-ahead threads to several cores. Future work to answer these questions will be valuable in CMP research.

8.2.3 More physical constraints

Three important physical constraints are investigated in this dissertation. As technology evolves, there will be more constraints calling for attention. There are at least two important constraints to which that future CMP work should pay attention. The first one is the reliability constraint. There are two kinds of reliability problems, both of which are growing more serious. The first one is called the lifetime reliability problem, dictated by the hard error rate due to wear-out based failures. The second one is called the soft error problem, due to transient faults arising from energetic particles in cosmic rays and alpha particles from packaging material. Many lifetime reliability failure mechanisms, like electro-migration, are exponentially dependent on temperature. As total heat flux and on-chip temperature become major problems in the future, so will lifetime reliability. The soft error rate is proportional to the number of transistors on chip. Exponentially increasing transistor counts will drive per-chip soft error rates up, causing it to pose an increasingly serious problem in the future. These reliability constraints are related to other constraints like chip area and chip temperature. How to incorporate them into the current infrastructure and how they will affect the results of CMP design space exploration is an open question. Furthermore, both CMP and SMT can be used as architectural infrastructures to provide redundancy and improve chip reliability, and it would be interesting to evaluate these techniques in my infrastructure, which considers almost all fundamental design dimensions of CMP.

Another important physical constraint which this dissertation does not consider is the parameter variation. Variability during manufacturing and during runtime are projected to increase. Within die

variation may dominate in the future. The obvious result of this trend is that even a CMP chip might be designed to be homogeneous, it could actually be heterogeneous in reality [33], due to variance in the performance and power of the cores. Adding this constraint to the current CMP design space exploration infrastructure and considering how to mitigate this problem in CMP architectures are challenging research problems.

Appendix A

Glossary

Definitions are collected here for easy reference. In general, the accepted definitions for terms are used, although some terms are used in a more restricted sense than their usual interpretation.

FO4 Fanout of 4. In an inverter chain when each succeeding inverter has a gate capacitance 4 times larger than that of the previous inverter, the delay of that chain is near the minimum for typical self-loading cases. Using units of FO4 is a way of referring to delay in a process-independent way.

Pipeline Depth The logic depth for one pipeline stage between two adjacent pipeline latches. Usually denoted as multiple of FO4. A bigger FO4 means a shallow pipeline.

Pipeline Width The number of execution pipelines in a superscalar chip.

Out-of-order Execution Out-of-order execution is an architecture technique that is used in many microprocessors in order to make use of cycles that would otherwise be wasted. Out-of-order execution can issue an instruction when its dependence is satisfied even if a previous instruction in program order is still waiting for execution.

In-order Execution In-order execution blocks issue if a previous instruction in program order is stalled and waiting for execution.

Turandot Turandot is a validated architecture level cycle accurate simulator for an IBM POWER4-like architecture.

Zauber Zauber is a CMP simulator that performs interpolation on traces provided by the core simulators. Zauber decouples detailed core simulation and the simulation of core interaction. It is much faster than detailed CMP simulators.

Bibliography

[1] A. G. Aipperspach, D. H. Allen, D. T. Cox, N. V. Phan, and S. N. Storino. 1.8-v, soi, 550-mhz, 64-b PowerPC microprocessor with copper interconnects. *IEEE Journal of Solid-State Circuits*, 1999.

[2] R. Alverson, D. Callahan, D. Cummings, et al. The Tera Computer System. pages 1–6, June 1990.

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

[4] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. Web search for a planet: The google cluster architecture. *IEEE Micro*, 23(2):22–28, April 2003.

[5] J. M. Borkenhagen, R. J. Eickmeyer, R. N. Kalla, and S. R. Kunkel. Multithreaded PowerPC processor for commercial servers. *IBM J. Res. and Dev.*, 44(6), November 2000.

[6] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of R & D*, 47(5/6), 2003.

[7] D. Brooks et al. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.

[8] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 83–94, June 2000.

[9] David Brooks, Pradip Bose, Viji Srinivasan, Michael Gschwind, Philip G. Emma, and Michael G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6), 2003.

[10] J. Burns and J.-L. Gaudiot. Area and system clock effects on SMT/CMP processors. In *Proceedings of the 2001 IEEE/ACM/IFIP International Conference on Parallel Architectures and Compilation Techniques*, pages 211–18, Sep. 2001.

[11] James Burns and Jean-Luc Gaudiot. SMT layout overhead and scalability. *IEEE Trans. Parallel Distrib. Syst.*, 13(2):142–155, 2002.

[12] P. Chaparro, J. González, and A. González. Thermal-aware clustered microarchitectures. In *Proceedings of the 2004 IEEE International Conference on Computer Design*, Oct. 2004.

[13] S. Chaudhry, P. Caprioli, S. Yip, and M. Tremblay. High performance throughput computing. *IEEE Micro*, 25(3):32–45, May/June 2005.

[14] J. Clabes et al. Design and implementatin of the Power5 microprocessor. In *ISSCC Digest of Technical Papers*, pages 56–57, Feb. 2004.

[15] John D. Davis, James Laudon, and Kunle Olukotun. Maximizing CMP throughput with mediocre cores. In *Proceedings of the 2005 IEEE/ACM/IFIP International Conference on Parallel Architectures and Compilation Techniques*, August 2005.

[16] D.Koufaty and D. T. Marr. Hyperthreading technology in the netburst microarchitecture. *IEEE Micro*, 23(2):56–65, 2003.

[17] J. Donald and M. Martonosi. Temperature-aware design issues for SMT and CMP architectures. In *Proceedings of the 2004 Workshop on Complexity-Effective Design*, June 2004.

[18] J. Donald and M. Martonosi. Techniques for multicore thermal management: Classification and new exploration. In *The 33rd Annual International Symposium on Computer*, Jun. 2006.

[19] James Dundas and Trevor Mudge. Improving data cache performance by pre-executing instructions under a cache miss. In *Proc. of ICS'97*, 1997.

[20] M. Ekman and P. Stenstrom. Performance and power impact of issue-width in chip-multiprocessor cores. In *Proceedings of the International Conference on Parallel Processing*, pages 359–68, Oct. 2003.

[21] A. El-Moursy, R. Garg, D. Albonesi, and S. Dwarkadas. Partitioning multi-threaded processors with a large number of threads. In *Proceedings of the 2005 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 112–23, Mar. 2005.

[22] M. Gowan, L. Biro, and D. Jackson. Power considerations in the design of the Alpha microprocessor. In *Proceedings of the 1998 Design Automation Conference*, Jun. 1998.

[23] G. Hamerly, E. Perelman, J. Lau, and B. Calder. Simpoint 3.0: Faster and more flexible program analysis. In *Proc. of the workshop on Modeling, Benchmarking and Simulation*, Jun. 2005.

[24] L. Hammand, B. A. Nayfeh, and K. Olukotun. A single-chip multiprocessor. *IEEE Computer*, 30(9):79–85, Sep. 1997.

[25] A. Hartstein and T. R. Puzak. The optimum pipeline depth for a microprocessor. In *Proceedings of the 29th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 7–13, May 2002.

[26] S. Heo, K. Barr, and K. Asanovic. Reducing power density through activity migration. In *Proceedings of the 2003 ACM/IEEE International Symposium on Low Power Electronics and Design*, Aug. 2003.

[27] K. R. Hirst, J. W. Haskins, Jr., and K. Skadron. dMT: Inexpensive throughput enhancement in small-scale embedded microprocessors with differential multithreading. *IEE Proceedings on Computers and Digital Techniques*, 151(1):43–50, Jan. 2004.

[28] M. S. Hrishikesh, D. Burger, N. P. Jouppi, S. W. Keckler, K. I. Farkas, and P. Shivakumar. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *Proceedings of the 29th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 14–24, May 2002.

[29] Z. Hu, D. Brooks, V. Zyuban, and P. Bose. Microarchitecture-level power-performance simulators: Modeling, validation, and impact on design. In *Tutorial at the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec. 2003.

[30] W. Huang, M. R. Stan, K. Skadron, S. Ghosh, K. Sankaranarayanan, and S. Velusamy. Compact thermal modeling for temperature-aware design. In *Proceedings of the ACM/IEEE 41st Design Automation Conference*, pages 878–883, June 2004.

[31] W. Huang, Mircea R. Stan, K. Skadron, K. Sankaranarayanan, S. Ghoshy, and S. Velusamy. Compact thermal modeling for temperature-aware design. Technical Report CS-2004-13, University of Virginia Department of Computer Science, Apr. 2004.

[32] J. Huh, D. Burger, and S. W. Keckler. Exploring the design space of future CMPs. In *Proceedings of the 2001 IEEE/ACM/IFIP International Conference on Parallel Architectures and Compilation Techniques*, pages 199–210, Sep. 2001.

[33] E. Humenay, D. Tarjan, and K. Skadron. Impact of parameter variations on multi-core chips. In *Proceedings of the 2006 Workshop on Architectural Support for Gigascale Integration, in conjunction with the 33rd International Symposium on Computer Architecture (ISCA)*, Jun. 2006.

[34] J.D.Collins, D.M.Tullsen, H.Wang, and J.P.Shen. Dynamic speculative precomputation. In *MICRO'01*, 2001.

[35] R. Kalla, B. Sinharoy, and J. Tendler. Power5: IBM's next generation power microprocessor. In *Proc. 15th Hot Chips Symp*, pages 292–303, August 2003.

[36] T. Karkhanis and J. E. Smith. A first-order superscalar processor model. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Computer Architecture*, June 2004.

[37] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. Comparing Power Consumption of SMT DSPs and CMP DSPs for Mobile Phone Workloads. In *International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES 2001)*, Nov. 2001.

[38] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-way multithreaded Sparc processor. *IEEE Micro*, 25(2):21–29, Mar./Apr. 2005.

[39] K. Krewell. UltraSPARC IV mirrors predecessor: Sun builds dual-core chip in 130nm. *Microprocessor Report*, pages 1, 5–6, Nov. 2003.

[40] B. Kumar and E. S. Davidson. Computer system design using a hierarchical approach to performance evaluation. *Communications of the ACM*, 23(9), Sept. 1980.

[41] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 81–92, Dec. 2003.

[42] R. Kumar, D. Tullsen, P. Ranganathan, N. P. Jouppi, and K. Farkas. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Computer Architecture*, June 2004.

[43] R. Kumar, D. M. Tullsen, and N. P. Jouppi. Core architecture optimization for heterogeneous chip multiprocessors. In *Proceedings of the 2006 IEEE/ACM/IFIP International Conference on Parallel Architectures and Compilation Techniques*, Sep. 2006.

[44] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Computer Architecture*, pages 64–75, June 2004.

[45] R. Kumar, V. Zyuban, and D. M. Tullsen. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *The 32nd Int'l Symp. on Computer Architecture*, June. 2005.

[46] S. R. Kunkel, R. J. Eickemeyer, M. H. Lipasti, T. J. Mullins, B. O.Krafka, H. Rosenberg a nd S. P. Vander-Wiel, P. L. Vitale, and L. D. Whitley. A performance methodology for commercial servers. In *IBM J. of Research and Development, 44(6)*, 2000.

[47] B. Lee and D. Brooks. Effects of pipeline complexity on SMT/CMP power-performance efficiency. In *Proc. of the Workshop on Complexity Effective Design*, Jun. 2005.

[48] H. Li, S. Bhunia, Y. Chen, T. N. Vijaykumar, and K. Roy. Deterministic clock gating to reduce microprocessor power. In *Proceedings of the Ninth IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2003.

[49] J. Li and J. F. Martínez. Power-performance implications of thread-level parallelism on chip multiprocessors. In *Proceedings of the 2005 IEEE International Symposium on Performance Analysis of Systems and Software*, pages 124–34, Mar. 2005.

[50] J. Li and J. F. Martínez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *Proceedings of the Twelfth IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2006.

[51] Y. Li, M. Hempstead, P. Mauro, D. Brooks, Z. Hu, and K. Skadron. Power and thermal effects of SRAM vs. Latch-Mux design styles and clock gating choices. In *Proceedings of the 2005 ACM/IEEE International Symposium on Low Power Electronics and Design*, Aug. 2005.

[52] Y. Li, Z. Hu, D. Brooks, K. Skadron, and P. Bose. Understanding the energy efficiency of simultaneous multithreading. In *Proceedings of the 2004 ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 44–49, Aug. 2004.

[53] Y. Li, B. C. Lee, D. Brooks, Z. Hu, and K. Skadron. CMP design space exploration subject to physical constraints. In *Proceedings of the Twelfth IEEE International Symposium on High-Performance Computer Architecture*, Feb. 2006.

[54] 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*, Feb. 2005.

[55] Deborah T. Marr, Frank Binns, David L. Hill, Glenn Hinton, David A. Koufaty, J. Alan Miller, and Michael Upton. Hyper-threading technology architecture and microarchitecture. *Intel Technology J.*, 6(1):4–15, Feb 2002.

[56] M.K.Chen and K.Olukotun. The jrpm system for dynamically parallelizing sequential java programs. In *ISCA'03*, June 2003.

[57] M. Monchiero, R. Canal, and A. González. Design space exploration for multicore architectures: A power/performance/thermal view. In *Proceedings of the 20th International Conference on Supercomputing*, June 2006.

[58] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, pages 114–17, Apr. 1965.

[59] M. Moudgill, P. Bose, and J. H. Moreno. Validation of Turandot, a fast processor model for microarchitecture exploration. In *Proc. of IEEE Internatioanl Performance, Computing and Communications Conf.*, pages 451–457, February 1999.

[60] Mayan Moudgill, John-David Wellman, and Jaime H. Moreno. Environment for PowerPC microarchitecture exploration. *IEEE Micro*, 19(3):15–25, 1999.

[61] O. Mutlu, J. Stark, C. Wilkerson, and Y.N.Patt. Runahead execution: An alternative to very large instruction windows for out-of-order processors. In *Proc. of HPCA'03*, Feb 2003.

[62] M. D. Powell, M. Gomaa, and T. N. Vijaykumar. Heat-and-run: Leveraging SMT and CMP to manage power density through the operating system. In *Proceedings of the Eleventh In-*

ternational Conference on Architectural Support for Programming Languages and Operating Systems, Oct. 2004.

[63] K. Ramani, N. Muralimanohar, and R. Balasubramonian. Microarchitectural techniques to reduce interconnect power in clustered processors. In *Proc. of the Workshop on Complexity Effective Design*, June 2004.

[64] 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 International Conference on Supercomputing*, pages 196–206, June 2004.

[65] Y. Sazeides and T. Juan. How to compare the performance of two smt microarchitectures. In *Proc. of IEEE Int'l Symp. on Performance Analysis of Systems and Software*, Nov. 2001.

[66] J. Seng, D. Tullsen, and G. Cai. Power-sensitive multithreaded architecture. In *Proceedings of the 2000 International Conference on Computer Design*, pages 199–208, 2000.

[67] T. Sherwood, E. Perelman, G.Hamerly, and B. Calder. Automatically characterizing large scale program behavior. In *Proceeding of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[68] SIA. *International Technology Roadmap for Semiconductors*, 2001.

[69] K. Skadron. Hybrid architectural dynamic thermal management. In *Proceedings of the 2004 ACM/IEEE/EDAA/EDAC Design, Automation and Test in Europe Conference*, pages 10–15, Feb. 2004.

[70] K. Skadron, K. Sankaranarayanan, S. Velusamy, D. Tarjan, M. R. Stan, and W. Huang. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1):94–125, Mar. 2004.

[71] Allan Snavely and Dean M. Tullsen. Symbiotic jobscheduling for a simultaneous multithreaded processor. In *Proc. of the Ninth Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 234–244. ACM Press, Nov. 2000.

[72] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *Proceedings of the 29th Annual ACM/IEEE International Symposium on Computer Architecture*, pages 25–34, May 2002.

[73] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. Reducing power in high-performance microprocessors. In *Proceedings of the 1998 Design Automation Conference*, Jun. 1998.

[74] M. Tremblay, B. Joy, and K. Shin. A three dimensional register file for superscalar processors. In *Proc. of the 28th Hawaii Int'l Conf. on System Sciences*, 1995.

[75] 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 *Proceedings of the 23rd ACM International Symposium on Computer Architecture*, 1996.

[76] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proceedings of the 22nd Annual ACM/IEEE International Symposium on Computer Architecture*, pages 392–403, June 1995.

[77] V.Zyuban, D.Brooks, V.Srinivasan, M.Gschwind, P.Bose, P.Strenski, and P.Emma. Integrated analysis of power and performance for pipelined microprocessors. *IEEE Transactions on Computers*, 53(8), August 2004.

[78] N. Weste and D. Harris. CMOS VLSI design: A circuit and systems perspective. In *Addison-Wesley*, 2005.

[79] V. Zyuban. *Inherently lower-power high-performance superscalar architectures*. PhD thesis, Univ. of Notre Dame, Mar. 2000.

[80] V. Zyuban and P. Strenski. Unified methodology for resolving power-performance tradeoffs at the microarchitectural and circuit levels. In *Proc. of ISLPED'02*, August 2002.