Quantifying Latency and Throughput Compromises in CMP Design

UNIVERSITY OF VIRGINIA DEPT. OF COMPUTER SCIENCE TECHNICAL REPORT CS-2006-26

Yingmin Li[†], Kevin Skadron[†], Benjamin Lee[‡], David Brooks[‡]
[†] Dept. of Computer Science, University of Virginia
[‡] Division of Engineering and Applied Sciences, Harvard University
{yingmin,skadron}@cs.virginia.edu, {dbrooks,bclee}@eecs.harvard.edu

Abstract

Designers of chip multiprocessors will increasingly be called upon to optimize for a combination of design metrics under a variety of design constraints. The adoption of chip multiprocessors has also led to a shift in design metrics toward aggregate throughput and away from single thread latency. We examine the compromises between latency and throughput under various power, thermal, area, and bandwidth constraints to quantify the latency penalties of a purely throughput optimized design. We consider a large chip multiprocessor design space that includes core count, core complexity (pipeline dimensions, in-order versus out-of-order execution), and cache hierarchy sizes.

We demonstrate an approach to effectively assess trade-offs given a comprehensive core model, a set of optimization criteria, and a set of design constraints. We perform a number of case studies to evaluate these trade-offs, exposing significant single thread latency penalties when optimizing solely for throughput and neglecting other measures of performance. As single thread latency continues to be one of several design metrics, any choice to compromise latency should be well understood before implementation. Collectively, our results suggest single thread latency is still a design metric of importance given that optimizing throughput alone will significantly compromise latency. Furthermore, the case for simple, in-order cores should be taken with caution given this balanced view of performance.

1 Introduction

Microprocessor design experienced a fundamental paradigm shift when performance gains across design generations were no longer achieved by frequency scaling and instruction-level parallelism, but by thread-level parallelism with chip multiprocessors and shared on-chip caches. This emphasis on thread-level parallelism promises greater energy efficiency and throughput performance. As throughput increasingly becomes the primary design metric, however, we find chip multiprocessors implemented with multiple simple (e.g., in-order), heavily multi-threaded processor cores. Such systems maximize throughput by providing an unparalleled number of computational threads within a reasonable power envelope, but potentially compromise other design metrics such as per thread latency. To effectively assess these trends and trade-offs, the designer must evaluate a large design space for a comprehensive set of design metrics and technology constraints.

The case for throughput computing is often supported by multiprocessor design exploration in a constrained design space. Techniques that decouple processor core simulation and interconnect modeling mitigate simulation costs and enable more comprehensive studies. Using such an approach, we evaluate a design space with varying core count, core complexity, and cache sizes. The most effective compromise between throughput and latency for designs within such a large space is not obvious, however, especially when evaluating these performance metrics in conjunction with other design metrics (e.g., power and temperature) and technology constraints (e.g., die area and pin bandwidth). Neglecting latency in throughput oriented design without quantifying the per thread latency penalties is a risky proposition as many applications remain

latency sensitive. For example, Internet service providers may increasingly differentiate themselves on the basis of response times or implement tiered service-level agreements with varying response time guarantees.

We consider a comprehensive chip multiprocessor design space that considers core count, core complexity, and cache hierarchy parameters (Section 3). Zauber, a methodology that decouples the simulation of individual cores from the simulation of chip fabric, enables the tractable evaluation of large number of core counts and designs (Section 4). We emphasize the need to consider this large design space in conjunction with a set of comprehensive design metrics and constraints. We demonstrate an approach to effectively assess trade-offs between these optimization criteria in a number of case studies (Section 5):

- **Design Space Characterization:** We quantify the trade-offs between single thread latency and aggregate throughput for various multiprocessor designs. This characterization exposes significant single thread latency penalties when optimizing solely for throughput and neglecting other measures of performance.
- **Sensitivity to Latency Constraints:** Given the importance of taking a holistic view of performance, we assess the sensitivity of the core design optimization to single thread latency constraints. We find simpler in-order cores often cannot meet more stringent latency constraints.
- **Sensitivity to In-Order Area:** In-order cores tend to be smaller than out-of-order cores and we assess the sensitivity of the core design optimization to more ambitious area assumptions for in-order execution. We find that out-of-order designs offer comparable throughput for lower latencies relative to in-order designs even under the most optimistic in-order area reductions.
- **Sensitivity to Bandwidth Constraints:** As area per core decreases and the number of cores per chip increases, pin bandwidth becomes increasingly relevant. We assess the sensitivity of the core design optimization to various bandwidth constraints, finding significant off-chip bandwidth may be required to mitigate increased demands on the memory hierarchy from a large number of small cores.
- **Dynamic SMT Support:** We present a preliminary analysis to dynamically disengage support for simultaneous multi-threading and improve single thread latency. This analysis suggests dynamic SMT support allows flexible trade-offs between latency and throughput once an architecture has been designed.

Collectively, our findings suggest single-thread latency continues to be an important design metric to be included in any optimization criteria. Arguments for simple, in-order cores should be taken with caution given that such designs appear to emphasize only throughput, potentially at significant cost to latency.

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. [3] explore the design space for core type, number of cores, cache size, and degree of multi-threading, 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. [9] 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. [2] 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. 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.

Li et al. [15] 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. Their work focuses on single-threaded, multi-programmed SPEC workloads. Monchiero et al. [16] 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 [12] 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 [13], they go on to develop heuristics for dynamic adaption 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 vein, Donald and Martonosi [4] develop heuristics for scheduling threads on a CMP to minimize thermal throttling, while Powell et al. [18] instead propose a core-hopping approach on CMPs in response to imminent thermal throttling.

Huh et al. [8] 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 we do, that out-of-order cores are generally preferable because of their greater area efficiency. Ekman and Stenstrom [5] 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.

Kumar et al. [10] 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 we are modeling throughput-oriented workloads consisting of independent threads, we follow the example of Niagara [9] and employ a similar sharing strategy. In our experiments, each L2 cache bank is shared by four cores. Interconnection design parameters are not variable in our design space at this time, and in fact constitute a sufficiently expansive design space of their own that we consider this to be beyond the scope of the current paper.

The methodologies for analyzing pipeline depth and width build on prior work by Lee and Brooks [11] by developing first-order models for capturing changes in core area as pipeline dimensions change, thereby enabling power density and temperature analysis. We identify optimal pipeline dimensions in the context of CMP architectures whereas most prior pipeline analyses consider single-core microprocessors [6, 7, 19]. 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 [20].

3 Design Space Models

3.1 Design Space

We model the configurations of Table 1 drawn from the design space specified by Table 2. In addition to considering issue width, data cache size, and issue style (i.e., in-order, out-of-order), we vary pipeline depth and L2 cache size for each configuration to optimize throughput under various constraints. Performance, power, and area for varying pipeline depth and width are estimated using prior models by Li, et. al. [15]. In particular, latencies scale linearly with the number of fan-out-of-four (FO4) delays per pipeline stage. A number of configurations implement 2-way simultaneous multi-threading modeled by increasing microarchitectural resources by 1.5X and scaling associated unconstrained power in prior work by Li, et. al. [14].

3.2 In-order Performance

In-order architectures (IO) differ from out-of-order architectures (OO) in their treatment of instruction dependencies. IO architectures block issue if a previous instruction is stalled due to an instruction dependency. We model issue logic to enforce in-order execution. Specifically, if an instruction is waiting for operands, all following instructions in program order will be blocked. For contrast, instructions will not block on the an instruction waiting for operands in out-of-order execution. Although we enforce in-order issue, we allow out-of-order retirement. Instructions in modern processors tend to

Configuration	004SMT	OO4	OO2	OS4	OS2	IO4SMT	IO4	IO2	IS4	IS2
Microarchitectural Summary										
Fetch/Issue Width	4/8	4/8	2/4	4/8	2/4	4/8	4/8	2/4	4/8	2/4
D-Cache Size	32KB			8KB		32KB			8KB	
Execution	out-of-order					in-order				
Area Summary										
Area (19FO4,mm^2)	11.22	8.38	8.68	6.45	12.89	9.85	7.46	7.31	5.52	10.61

Table 1. Design Summary

Microarchitectural Parameters	
Execution	in-order, out-of-order
Depth	18-36FO4, steps of 6FO4
Width	4,8 issue bandwidth
Register File	$\{$40GPR, 36FPR$\}_{in-order}$, $\{$80GPR, 72FPR$\}_{out-of-order}$
Issue Queues	$\{$Issue Width$\}_{in-order}$, $\{$40FXU, 10FPU, 36LSU, 12BR$\}_{out-of-order}$
L1 I-Cache	32KB
L1 D-Cache	2-128KB, factors of 4
L2 Cache	2-8MB, factors of 2

Table 2. Microarchitectural Parameters

have very different execution latencies and out-of-order retirement can improve the performance of our in-order architecture with very little hardware overhead.

In addition to these changes to the fundamental pipeline logic, we reduce the sizes of microarchitectural resources from those in the original OO design. For the same issue width, we halve the IO sizes of the physical register, load/store queue, and retirement queue relative to OO sizes. We also set the IO issue queue size to the issue width since large issue queues are unnecessary due to in-order execution.

3.3 In-order Power

We need to scale down the unconstrained power (i.e., power before clock gating effects are considered) for all resources whose sizes are changed relative to the OO model. We use CACTI [21] to calculate the scaling factor for the L1 caches and first-level TLB's. Changes in pipeline width will impact power attributed to functional units and SRAM arrays (via additional ports). We assume unconstrained hold and switching power increases linearly with the number of functional units and access ports, a valid assumption since we model an architecture with clustered functional units and replicated caches for additional read ports [11]. For certain structures, linear power scaling may be optimistic and, for example, does not capture non-linear relationships between power and the number of register file access ports since it does not account for the additional circuitry required in a multi-ported SRAM cell. For this reason, we apply superlinear power scaling with exponents drawn from Zyuban's work in estimating energy growth parameters [22]. Since these parameters were experimentally derived through analysis of a non-clustered architecture and we model a clustered architecture, we only apply this power scaling to the non-clustered components of our architecture.

Our power models do not consider the power benefits of decreasing hardware complexity for certain structures when designing for IO instead of OO cores. For example, issue queues in an IO architecture can be much simpler due to simpler wake-up and issue logic. However, the size of those resources in the IO model is very small (four to eight entry issue queues) and power scaling to account for these effects will have little impact on the estimate of total power consumption.

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

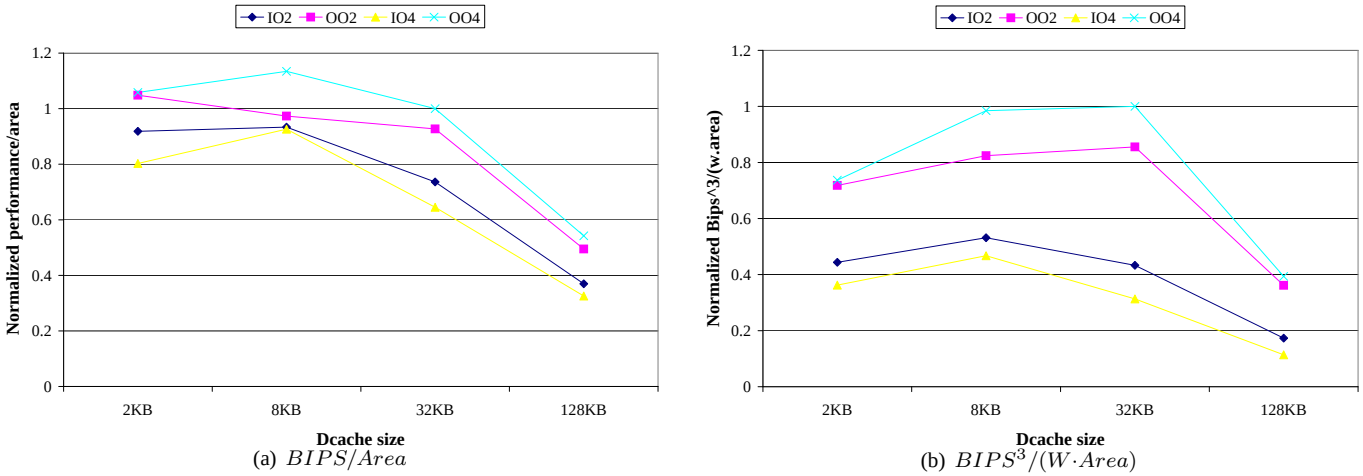
Table 3. Baseline Latencies

3.3.1 In-order Area

Table 1 presents the area estimates in 65nm technology for architectures in our design space. As in the power models, we scale down the area for all resources whose sizes are changed relative to the OO model. Area estimates for the L1 caches and first-level TLB's are obtained from CACTI. We assess the sensitivity of our findings to the area model in Section 5.3.

The core area may occupy less than 50% of the chip area in chip multiprocessors and it is important to model the area of other on-chip structures, including the L2 cache ($11.52mm^2$), DDR/DRAM controllers ($0.96mm^2$), and interconnect networks ($11.40mm^2$). We estimate the on-chip L2 cache area from the IBM POWER4 die photo, scaling it to 65nm technology. Our area estimates for DDR/DRAM controllers are based on the Sun Niagara die photo and its total chip memory bandwidth. We assume a linear area scaling of DDR/DRAM controllers relative to the chip's maximum pin bandwidth. Finally, we assume every four cores share an L2 cache, and model a 4-way crossbar implemented in a high metal layer as in previous work by Kumar et. al. [10].

The L1 cache structure may be responsible for more than half the core area and therefore properly selecting the size for these structures is critical to chip area efficiency. We sweep several L1 cache sizes, adjusting all related on-core cache structures proportionally, and evaluate performance and power-performance efficiency ($BIPS^3/W$). As shown in Figure 1, the best $BIPS/Area$ and $BIPS^3/(W \cdot Area)$ are achieved with 8KB or 32KB L1 data caches for almost all architectures. Therefore, we perform all following experiments with these two L2 cache sizes.



(a) $BIPS/Area$ (b) $BIPS^3/(W \cdot Area)$

Figure 1. $BIPS/Area$ **and** $BIPS^3/(W \cdot Area)$ **of varying L1 data cache sizes.**

4 Experimental Methodology

4.1 Simulation Framework

We employ Zauber to tractably simulate a large number of cores and core designs, a simulation methodology that decouples the simulation of individual cores from the simulation of chip fabric and L2 cache [15]. We first use Turandot/PowerTimer, a cycle-accurate trace-driven simulator to collect core performance and power characteristics for 10K instruction segments [1, 17]. Turandot/PowerTimer originally modeled a POWER4-like out-of-order architecture, but has been extended for design space exploration. The simulator has also been modified to generate traces of single-core L2 cache access patterns. Zauber, a shared L2 cache and fabric simulator, uses these access patterns to interpolate performance and power results for various chip multiprocessor configurations. By separately simulating the L2 and fabric, Zauber accelerates CMP simulation. We also use a simplified temperature model validated against Hotspot 2.0 to estimate thermal effects at core granularity [15].

4.2 Design Metrics and Constraints

We evaluate the design space for performance, differentiating between single thread latency and aggregate throughput. We also evaluate designs for power and thermal efficiency by comparing the performance of chips with heatsinks of varying effectiveness. In addition to these optimization criteria, we impose hard physical constraints on the total chip power (250W) and peak temperature (100 C). If these constraints are reached, we use voltage scaling to throttle chip speed and ensure these constraints are satisfied. We also consider two different area constraints (200, 400 mm^2) corresponding to different CMP markets. We examine three different pin bandwidth constraints (24, 48, 96 GB/s). As with power and temperature, if the chip reaches pin bandwidth limitations, the chip is throttled to ensure average bandwidth does not exceed maximum available bandwidth. We do not consider the burdens of bursty traffic on chip pin bandwidth in this work.

4.3 Benchmarks Methodology

We use two benchmarks (JBB, MCF) for case studies to demonstrate our approach for quantifying trade-offs between single-thread latency and aggregate throughput under various physical constraints in a comprehensive design space. We do not leverage any particular feature of these benchmarks and our approach is generally applicable. These single-threaded applications are representative of workloads that may require trade-offs between latency and throughput since they exhibit task-level parallelism but may still require low single-thread latency.

- **JBB** represents an e-commerce workload. Certain markets or customers may require guaranteed response times in tiered service-level agreements. SpecJBB is interesting because it is insensitive to L1 cache size and produces significant off-chip traffic regardless of L2 cache size. This benchmark also benefits from the instruction level parallelism of OO cores.
- **MCF** computes a minimum-cost-network for traffic routing and is representative of decision support applications that may require interactive responses or iteratively refine their answer, delivering the best available answer within a given time. The MCF benchmark has a 2MB working set and is thus extremely memory bound unless 2MB of L2 cache per thread is available.

5 Results

We take *single thread latency* as the delay for a billion instructions of a particular thread. In contrast, we take *aggregate delay* as the delay for a billion instructions across all thread contexts. Aggregate delay is equivalent to the inverse of aggregate throughput measured in BIPS. We first characterize the latency and throughput trade-offs of designs in Table 1 when optimizing for throughput and neglecting latency. Introducing latency into the optimization, we then maximize throughput for a range of latency targets. Recognizing the recent trend toward simpler, smaller cores, we assess the sensitivity of our results to smaller in-order core areas. However, as area per core decreases and the number of cores per chip increases,

Configuration	Cores	Depth (FO4)	L2 Cache (MB)	Voltage (V)
OS2	28	24	2	1.00
OS2SMT	20	24	8	0.97
OS4	20	36	8	1.00
OS4SMT	16	30	8	0.96
OO2	24	24	2	1.00
OO2SMT	20	30	4	1.00
OO4	20	30	2	1.00
OO4SMT	16	36	4	1.00
IS2	28	24	4	1.00
IS2SMT	24	18	4	0.90
IS4SMT	20	30	8	0.97
IO2	24	24	4	1.00
IO2SMT	20	18	4	0.92
IO4SMT	20	30	4	0.98

Table 4. Throughput Optimized Designs of Figure 2

pin bandwidth becomes increasingly relevant and we will examine the sensitivity of our results to bandwidth constraints. Lastly, we assess the potential of dynamic simultaneous multi-threading (SMT) capabilities in which a core design for SMT restricts itself to one thread.

5.1 Design Space Characterization

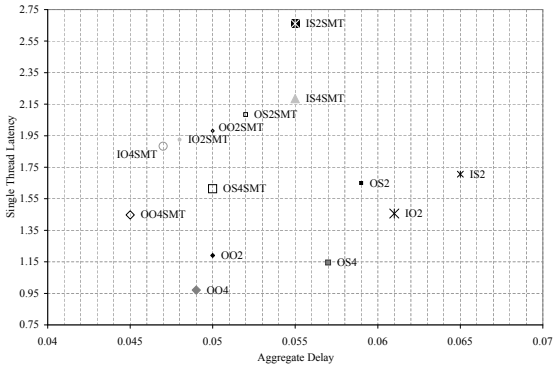


Figure 2. [JBB] :: Single thread latency versus aggregate delay (inverse throughput).

Figure 2 plots JBB single thread latency against aggregate throughput for the architectures of Table 1. Each point in this figure displays the throughput optimized configuration chosen from various core counts, pipeline depths, L2 cache sizes, and voltage scaling levels identified by Table 4 to meet thermal constraints. This figure shows that OO configurations tend to have both the lowest single thread latency and the lowest aggregate delay (i.e., highest aggregate throughput). This trend is partly because OO cores are designed with relatively shallow pipelines and modest L2 caches, but primarily because the OO architectures we consider inherently achieve better performance and $BIPS^3/(W \cdot Area)$. For both JBB and MCF, such architectures are an area efficient way to improve the number of instructions executed per cycle.

Introducing SMT improves throughput for both OO and IO designs. Holding the number of simultaneous threads constant at 2 while increasing superscalar width from 2 to 4 will generally improve throughput for OO configurations as more

resources are supplied for the same demand. However, increasing the number of threads per pipeline does not necessarily help IO cores because such cores do not favor wider pipelines as a means to exploit instruction level parallelism. For relatively narrow pipelines, we find the throughput gains of simultaneous multi-threading are achieved at the expense of single thread latency. For example, the optimal IS2 design achieves aggregate delay and single thread latency of 0.065 and 1.706 secs/billion instructions. In contrast, the optimal IS2SMT design achieves delay and latency of 0.055 and 2.661. In effect, multi-threading for IS2 designs will reduce aggregate delay by 15% while increasing single thread latency by 56%. Similarly, multi-threading for IO2 designs will reduce aggregate delay by 21% while increasing individual thread latency by 32%.

5.2 Sensitivity to Latency Constraints

As shown by the previous characterization of single thread latency and aggregate delay (inverse throughput), significant latency penalties may by incurred if latency is neglected when optimizing for throughput. For this reason, the optimization criteria should include particular latency targets. Specifically, we consider the sensitivity of the optimization problem given latency constraints by restricting candidate solutions to those with single thread latencies within $n\%$ of a previous generation design. We take our POWER4-like baseline as our reference and sweep n from 10% to 90%. For each design in Table 1, we optimize for total throughput after meeting the latency constraint. Note that some designs are not able to achieve the latency target and we omit their results. After optimizing core count, pipeline depth, L2 cache size, and voltage scaling to maximize throughput for a latency target, we again consider our two performance metrics: single thread latency and aggregate delay (i.e., inverse throughput).

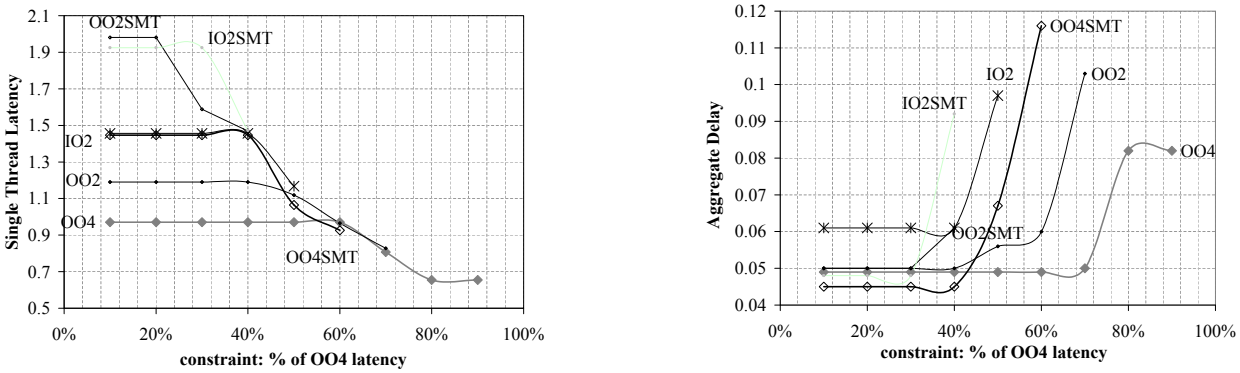


Figure 3. [JBB,$400mm^2$,LR] :: Single thread latency (L) and aggregate delay (R) versus latency constraint

We first present results for SpecJBB with both the low-resistance (LR) and high-resistance (HR) heatsinks (Figures 3–4). Figure3L shows the single-thread latency for each architectural configuration. For $n > 50\%$ 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 OO4SMT configuration is only able to meet the 60% latency constraint.

Figure3R shows the optimized aggregate delay (inverse throughput) for the designs of Figure 3L. The OO4SMT configuration achieves the best throughput up to the 40% constraint, after which the OO4 configuration achieves better throughput. The best in-order configuration, IO2SMT, is competitive with each of the other out-of-order architectures OO4SMT and OO4 up to the 30% constraint, but is otherwise unable to achieve both the low single thread latency and high aggregate throughput of the OO designs. This observation is also true for IO configurations without SMT. If target design requires single-thread latency closer to the previous generation (e.g. within 50%), then the OO4 configuration achieves the best throughput. If strict single thread latency requirements are enforced, all other architectures must sacrifice throughput for single thread latency (for example, by moving to a deeper pipeline or increasing L1 cache sizes) while the OO architecture

can maintain near peak throughput without compromising latency.

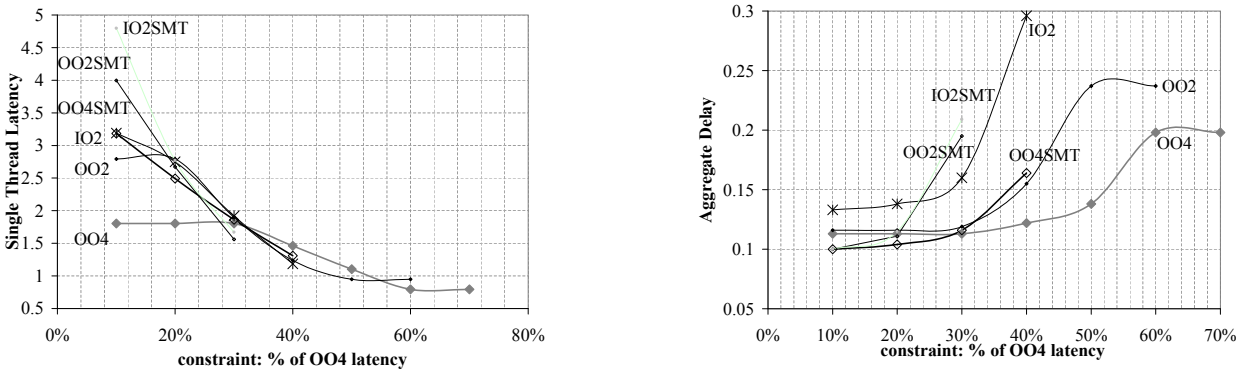


Figure 4. [JBB,$400mm^2$,HR] :: Single thread latency (L) and aggregate delay (R) versus latency constraint

Figure4L repeats the analysis after imposing harsher thermal constraints by replacing the low-resistance heatsink with a high-resistance heatsink. Compared to the LR case, many more configurations are eliminated from consideration as they violate the latency targets as the apply severe DVFS throttling in efforts to meet thermal constraints. Although the reference POWER4-like architecture also uses a high-resistance heatsink, the increased core count in throughput optimized designs induces significant global heatup via the heat spreader to cause additional throttling. Overall, additional thermal constraints reduce the differences in single thread latency between IO and OO designs. As shown in Figure 4L, the IO2 design achieves the lowest single-thread latency at the 40% constraint and the IO2SMT design is very close to best at the 30% constraint. At the 50% latency constraint, all in-order designs are eliminated and OO2 design is favored for its superior power, and as a result, thermal characteristics relative to OO4. Thus, under severe thermal constraints simpler cores can achieve lower latency than the OO4 core.

Figure4R shows the aggregate throughput with the HR configuration. We find that the IO2SMT and the OO4SMT configuration are comparable, with both achieving aggregate delays of approximately 0.115 seconds per billion instructions at the 10% and 20% latency constraints. With constraints less than 30% many configurations are quite close and OO4 is only clearly better when the constraint exceeds 40%.

Figures 5–6 present similar results for MCF. Overall, the trends observed for JBB are also valid for MCF. exhibiting a wide spread in single-thread latency between IO and OO configurations. When considering aggregate delay (inverse throughput), the OO SMT configurations are clearly the best choices for the smaller latency constraints ranging from 10% to 40%. We observe IO architectures generally cannot achieve the latency targets greater than 60% of baseline. For latency targets less than 60%, OO designs achieves lower aggregate delay and higher aggregate throughput relative to IO designs. Mcf is a memory bound benchmark and tends to favor large L2 caches in the optimal configuration. This further weakens the area advantage of IO architectures as the L2 cache occupies a larger portion of the chip area. Given the smaller area budget for processing cores, in-order designs cannot exploit their low area as effectively, placing additional cores for greater throughput. Thus, chips with larger caches will favor OO over IO designs for a given area budget.

5.3 Sensitivity to In-Order Area

In-order cores tend to be smaller than out-of-order cores as several resources supporting out-of-order execution (e.g., register rename tables, issue queues) become unnecessary. Reducing the area of other non-core on-chip structures will further improve the relative area advantage of IO designs. This IO area advantage translates into an increased core count for a given area budget that favors aggregate throughput. Recognizing the recent trend toward simpler, smaller cores, we assess the sensitivity of our results to more aggressive estimates of in-order core area. First, the interconnection area and power are reduced to 10% of the original assumption. We then consider IO core area reductions of 10%, 30%, and 50% relative to

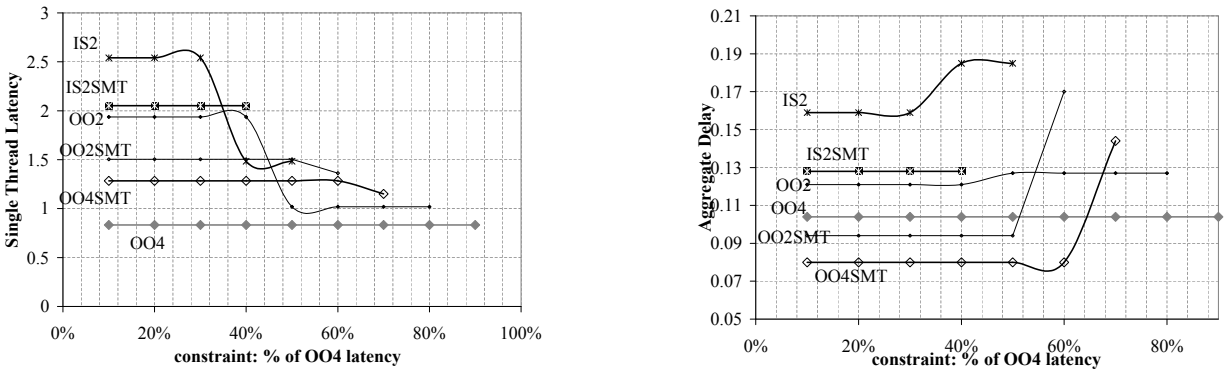


Figure 5. **[MCF,$400mm^2$,LR] :: Single thread latency (L) and aggregate delay (R) versus latency constraint**

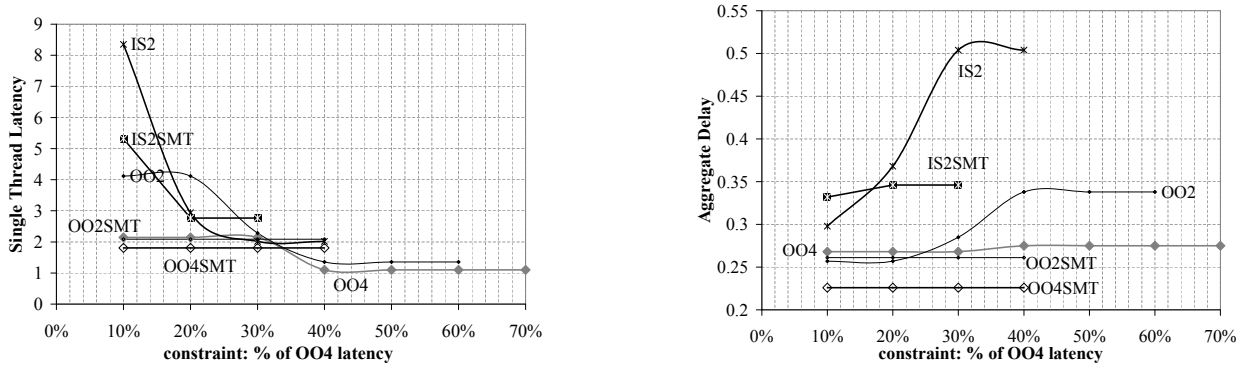


Figure 6. **[MCF,$400mm^2$,HR] :: Single thread latency (L) and aggregate delay (R) versus latency constraint**

the original area estimates while keeping OO core area estimates unchanged. We constrain the total chip area to 200 mm^2 and pin bandwidth to 48GB/s.

Figure 7 presents the improvements in JBB aggregate delay (inverse throughput) resulting from smaller per core area and a greater number of cores per chip. As the reduced area estimates become increasingly optimistic (50% to 90%), we observe aggregate delay reductions of up to 40%. For example, Figure 7L indicates, for a low-resistance heatsink, aggregate delay falls 41% from 0.081 to 0.048 and and 21% from 0.053 to 0.042 seconds per billion instructions for IO2 and IO2SMT, respectively. Similarly for a high-resistance heatsink of Figure 7R, aggregate delay falls 42% and 41% for IO2 and IO2SMT. The high-resistance heatsink favors the power and thermal characteristics of IO cores. For example, IO2SMT cores with 70% and 90% area reductions achieve 22% and 44% lower aggregate delay relative to OO4SMT, the highest throughput OO core. However, in nearly all cases, the throughput optimal IO designs exhibit worse single thread latency than OO designs.

In contrast, Figure 8 indicates that even if we assume 50% IO area reductions, the throughput maximizing IO configuration is only comparable to the best OO configuration. Aggregate delays for IS2 and OO4SMT differ by only 5% or 6% regardless of heatsink effectiveness. MCF is a memory bound workload and tends to favor larger L2 caches in its optima. Thus, the area advantages of IO designs are limited as core area budgets are constrained to favor larger cache area budgets. Illustrating this effect, the throughput of IS2SMT in Figure 8L is unchanged despite area scaling from 90% to 50%. Adding

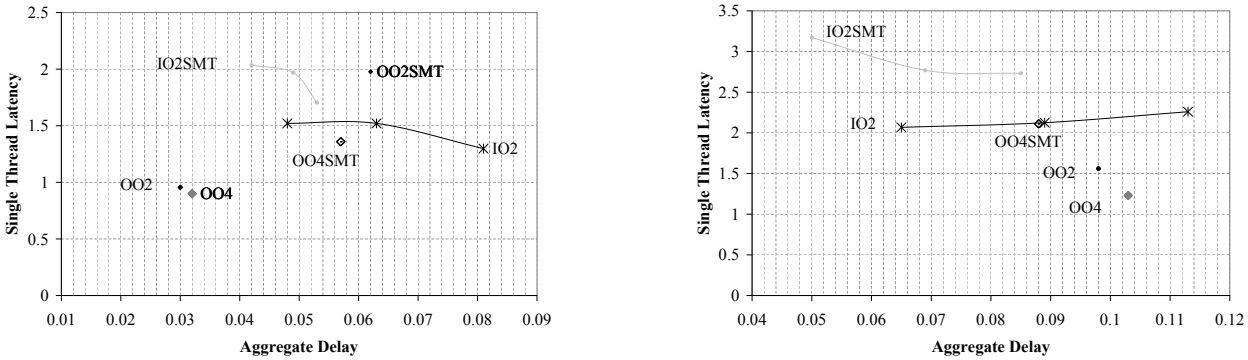


Figure 7. [JBB] :: Single thread latency, aggregate throughput sensitivity to area reductions with LR (L) and HR (R) heatsinks

cores while holding L2 cache sizes constant results in much higher cache miss rates, negative aggregate throughput return, and higher pin bandwidth requirements. Addressing these bottlenecks by increasing L2 cache sizes to hold working sets from additional cores will exceed the chip area constraint. More severe thermal constraints could favor IO architectures as shown in Figure 8R. Even in this case, only IS2 with the most optimistic area reductions achieves lower aggregate delay and greater throughput than the best OO design (5% difference from OO4SMT).

5.4 Sensitivity to Bandwidth Constraints

As area per core decreases and the number of cores per chip increases, pin bandwidth becomes an increasingly relevant challenge for multiprocessor designers. We assess the sensitivity of our results to pin bandwidth constraints. We model increased pin bandwidth by increasing the total number of DDR channels on the die and consider total chip bandwidth of 24, 48 or 96 GB/s. Since these additional channels require additional area that may restrict the number of cores or L2 cache on the chip, more pin bandwidth can actually be detrimental to total throughput depending on the exact trade-offs.

Figure 9 presents the single thread latency and aggregate delay trade-offs under pin bandwidth constraints for JBB and MCF. The 96GB/s point is identified for each design. 48 and 24GB/s are identified by moving along connecting lines. In several cases, additional pin bandwidth is essential. For example, OO4SMT of Figure 9L requires additional pin bandwidth reduces latency while increasing total throughput and 96GB/s maximizes the overall performance of JBB on this design. In other cases, additional pin bandwidth appears less important as the performance benefits of additional bandwidth do not justify the additional area overhead of the associated DDR channels. For example, IO2SMT and OO2 achieve the lowest aggregate delays with 48GB/s. Figure 9R suggests pin bandwidth is less performance critical for MCF OO designs. For example, OO2 and OO4SMT achieve lower latency and greater throughput, respectively, with 24GB/s. In contrast, the IO2 designs require significant off-chip bandwidth to reduce single thread latency, mitigating increased demands on the memory hierarchy from a large number of small cores.

5.5 Dynamic SMT Support

While our results suggest simultaneous multi-threading (SMT) support is an efficient mechanism for achieve high throughput, we need to be cognizant of single thread latency. Dynamically making SMT available will enable greater single thread latency is valued over aggregate throughput. Figure 10 evaluates the latency and throughput trade-offs for turning off SMT support for an SMT design. Each line in the figure connects latency-throughput points of designs with SMT on

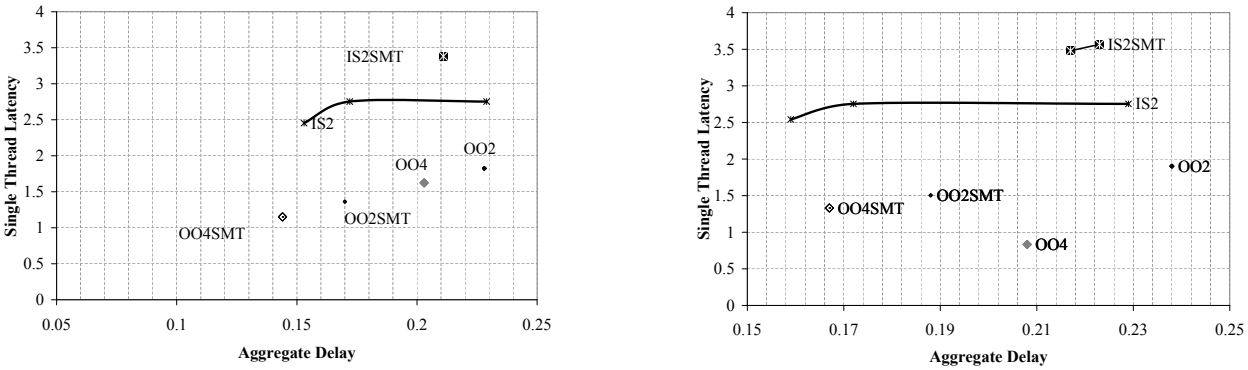


Figure 8. [MCF] :: Single thread latency, aggregate throughput sensitivity to area reductions with LR (L) and HR (R) heatsinks

(upper-left) and off(lower-right). Turning off SMT for an SMT core improves single thread performance between 21% (IO2SMT) and 43% (OO4SMT) for all architectures by eliminating pipeline resource contention. However, disengaging SMT support will incur aggregate throughput penalties between 15% (OS2SMT) and 31% (IO2SMT). Thus, dynamically turning off SMT support may be a viable technique to adjust the trade-off between single thread latency and aggregate throughput for a given architecture.

6 Conclusions

This work provides a comprehensive analysis of CMP design when considering performance in terms of both single-thread latency and aggregate chip throughput. We perform this analysis 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, out-of-order cores remain competitive for chip multiprocessors when the single thread latency penalties of simpler in-order cores are considered. More generally, we believe design space studies must increasingly take a balanced view of performance to include both latency and throughput. Maximizing throughput by providing a large number of computational threads will continue to be important, throughput must be optimized in conjunction with other design metrics and constraints. The recognition that not all applications are throughput oriented suggests designs may require a mixture of low single thread latency and high aggregate throughput. Our findings suggest out-of-order execution may still be the best approach to achieve this mixture.

References

[1] 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.

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

[3] J. D. Davis, J. Laudon, and K. 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.

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

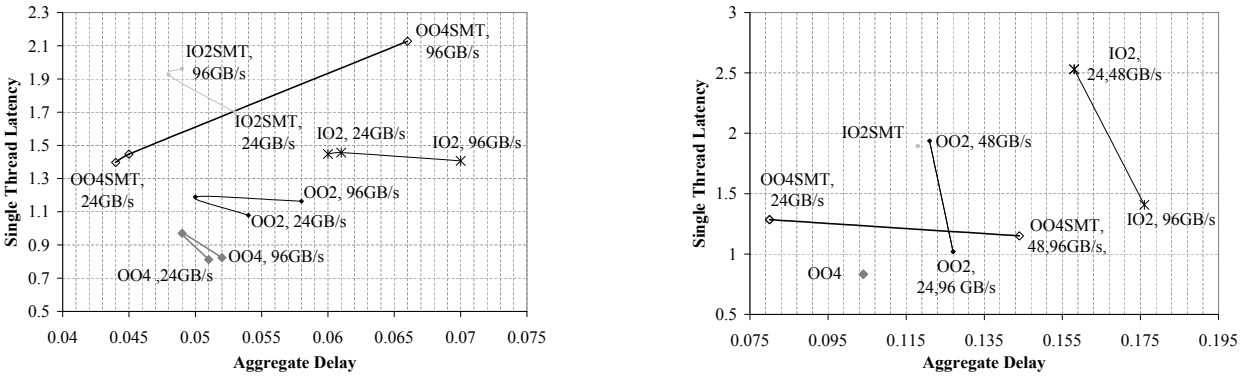


Figure 9. **[$400mm^2$,LR] :: Single thread latency, aggregate throughput sensitivity to pin bandwidth constraints for JBB (L) and MCF (R)**

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] M. Moudgill, P. Bose, and J. Moreno. Validation of Turandot, a fast processor model for microarchitecture exploration. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC)*, pages 451–457, Feb. 1999.

[18] 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 International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2004.

[19] 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.

[20] 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.

[21] S. J. E. Wilton and N. P. Jouppi. Cacti: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–88, May. 1996.

[22] V. Zyuban. *Inherently Lower-power High-performance Superscalar Architectures*. PhD thesis, Univ. of Notre Dame, Mar. 2000.

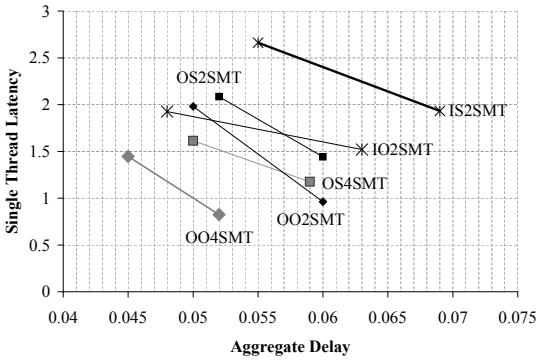


Figure 10. Dynamic SMT Capability