Chip Multithreading: Opportunities and Challenges

Lawrence Spracklen & Santosh G. Abraham
Scalable Systems Group
Sun Microsystems Inc., Sunnyvale, CA
{lawrence.spracklen,santosh.abraham}@sun.com

Abstract

Chip Multi-Threaded (CMT) processors provide support for many simultaneous hardware threads of execution in various ways, including Simultaneous Multithreading (SMT) and Chip Multiprocessing (CMP). CMT processors are especially suited to server workloads, which generally have high levels of Thread-Level Parallelism (TLP). In this paper, we describe the evolution of CMT chips in industry and highlight the pervasiveness of CMT designs in upcoming general-purpose processors. The CMT design space accommodates a range of designs between the extremes represented by the SMT and CMP designs and a variety of attractive design options are currently unexplored. Though there has been extensive research on utilizing multiple hardware threads to speed up single-threaded applications via speculative parallelization, there are many challenges in designing CMT processors, even when sufficient TLP is present. This paper describes some of these challenges including, hot sets, hot banks, speculative prefetching strategies, request prioritization and off-chip bandwidth reduction.

1 Introduction

Computer architects have achieved dramatic gains in single-thread performance, using a range of microarchitectural techniques, such as, superscalar issue, out-of-order issue, on-chip caching, and deep pipelines supported by sophisticated branch predictors. Each generation of process technology doubles the number of available transistors and, until recently, these additional resources have been harnessed toward improving single-thread performance.

However, a plot of delivered performance versus chip size (in transistors) demonstrates that the efficiency with which we are able to utilize the available transistor budget has declined over time. Power and memory latency considerations place additional obstacles to improving single-thread performance. While recent attempts at improving single-thread performance, through even deeper pipelines, have led to impressive clock frequencies, these clock frequencies have not translated into demonstrably better performance over less aggressive designs.

Server workloads are broadly characterized by high levels of thread-level parallelism (TLP), low instruction-level parallelism (ILP) and large working sets. The potential for further improvements in overall single-thread CPI is limited; on-chip CPI cannot be improved significantly due to low ILP and off-chip CPI is large and growing, due to relative increases in memory latency. However, typical server applications concurrently serve a large number of users or clients; for instance, a contemporary database server may have hundreds of active processes, each associated with a different client. Furthermore, these processes are currently multi-threaded to hide disk access latencies. This structure leads to high levels of TLP. Thus, it is extremely attractive to couple the high TLP in the application domain with support for multiple threads of execution on a processor chip.

Chip Multi-Threaded (CMT) processors support many simultaneous hardware strands (or threads) of execution via a combination of support for multiple cores (Chip Multiprocessors (CMP)) [1] and Simultaneous Multi-Threading (SMT) [2]. SMT combats increasing latencies by enabling multiple strands to share many of the resources within the core, including the execution resources. With each strand spending a significant portion of time stalled waiting for off-chip misses to complete, each strand's utilization of the core's execution resources is extremely low. SMT improves the utilization of key resources and reduces the sensitivity of an application to off-chip misses. Similarly, CMP enables multiple cores to share chip resources such as, the memory controller, off-chip bandwidth, and the L2 cache, improving the utilization of these resources. In our taxonomy, SMT and CMP are two extremes of a continuum characterized by varying degrees of sharing of on-chip resources among the strands.

Though the arguments for CMT processors are often made in the context of overlapping memory latencies, memory bandwidth considerations also play a significant role. Upcoming memory technologies such as Fully-Buffered DIMMS (FBD) have higher bandwidths (say 60 GB/s/chip), as well as higher latencies (say 130 ns), pushing up their bandwidth-delay product [3] to 60GB X 130 ns = 7800 bytes. The processor chip's pins represent an expensive resource and, in order to keep these pins fully utilized (assuming a cache line size of 64 bytes), the processor chip needs to sustain 7800/64 or over 100 parallel requests. A single strand on an aggressive out-of-order processor core generates less than two parallel requests on typical server workloads [4]: therefore, a large number of strands are required to sustain a high utilization of the memory ports.

Finally, power considerations also favor CMT processors. Given the almost cubic dependence between core frequency and power consumption [5], power consumption drops dramatically with reductions in frequency. As a result, for workloads with ad-

equate TLP, doubling the number of cores and halving the frequency delivers roughly equivalent performance, while reducing power consumption by a factor of four.

In this paper, we describe the evolution of CMT processors both within and outside Sun, the CMT design space and some of the challenges in designing effective CMT systems. Though CMT processors have been proposed and evaluated in academic research, a disproportionate fraction of the work has been devoted to speculative parallelization approaches that enable a CMT chip to deliver a speedup on single-threaded applications [6]. Our objective is to highlight the pervasiveness of CMT designs in industry, and attract academic research in some other important areas.

2 Evolution of CMTs

Given the exponential growth in transistors per chip over time, a rule of thumb is that a board design becomes a chip design in ten years or less. Thus, most industry observers expected that chip-level multiprocessing would eventually become a dominant design trend. The case for a single-chip multiprocessor was presented as early as 1996 by Olukotun's team [1]. The Stanford Hydra CMP processor design called for the integration of four MIPS-based processors on a single chip [7]. A DEC/Compaq research team proposed the incorporation of eight simple Alpha cores and a two-level cache hierarchy on a single chip and estimated a simulated performance of 3x that of a single-core next generation Alpha processor for on-line transaction processing workloads [8].

As early as the mid-1990s, Sun recognized the problems that would soon face processor designers as a result of the rapidly increasing clock frequencies required to improve single-thread performance. In response, Sun defined the MAJC architecture to target thread-level parallelism [9]. Providing well-defined support for both CMP and SMT processors, MAJC was industry's first step towards general-purpose CMT processors. Shortly after publishing the MAJC architecture, Sun announced its first MAJC-compliant processor (MAJC-5200), a dual-core CMT processor, with cores sharing an L1 data cache [10].

Since then, Sun has also begun to move its SPARC processor family towards the CMT design point. In 2003, Sun announced two CMT SPARC processors: Gemini, a dual-core UltraSPARC-II derivative [11] (as illustrated in Figure 1(a)) and UltraSPARC-IV (code named Jaguar), a dual-core UltraSPARC-III derivative [12]. These first-generation CMT processors were derived from earlier uniprocessor designs and the two cores do not share any resources, except for the off-chip data paths. In 2003, Sun also announced its second generation CMT processor, UltraSPARC-IV+, a follow-on to the current UltraSPARC-IV processor, in which the on-chip L2 and off-chip L3 caches are shared between the two cores, as illustrated in Figure 1(b) [12].

Sun also recently announced a 32-way CMT SPARC processor (code-named Niagara) [13], which is already running in the lab and is due for general availability in 2006. Niagara has eight cores and each core is a four-way SMT with its own private L1 caches. All eight cores share a 3MB, 12-way L2-cache, as illustrated in Figure 1(c). Niagara represents a third-generation CMT processor, where the entire design, including the cores, is optimized for a CMT design point. Since Niagara is targeted at commercial server workloads with high TLP, low ILP and large working sets, the ability to support many strands and therefore many concurrent off-chip

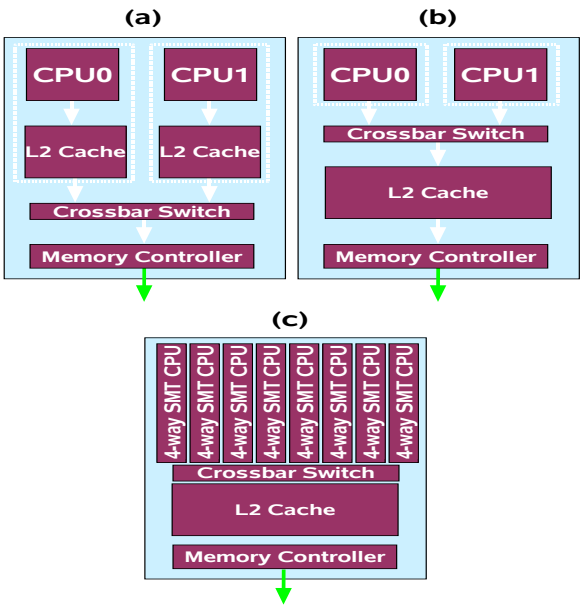

Figure 1. Overview of CMT design trends a) First generation CMTs, b) Second generation CMTs and c) Future third-generation CMTs

misses is key to overall performance. Thus, in order to accommodate 8 cores, each core supports single issue and has a fairly short pipeline.

Sun's move toward the CMT design has been mirrored throughout industry. In 2001, IBM introduced the dual-core POWER-4 [14] processor and recently released their second generation CMT processor, the POWER-5, in which each core supports 2-way SMT [15]. AMD, Fujitsu and Intel have also announced their intentions to release dual-core CMP processors in the near future [16, 17, 18]. While this fundamental shift in processor design was initially confined to the high-end server processors, where the target workloads are the most thread-rich, this change has recently begun to spread to desktop processors [19].

CMT is emerging as the dominant trend in general-purpose processor design, with manufacturers discussing their multi-core plans beyond their initial dual-core offerings [19]. Similar to the shift from CISC to RISC, that enabled an entire processor to fit on a single chip and internalized all communication between pipeline stages to within a chip, the move to CMT represents a fundamental shift in processor design that internalizes much of the communication between processors to within a chip.

3 CMT design space

An attractive proposition for evolving a CMT design is to just double the number of cores per chip every generation, because a new process technology essentially doubles the transistor budget. Little design effort is expended on the cores and performance is almost doubled every process generation on workloads with sufficient TLP. Many first generation CMT processor designs comprise of two "carbon-copies" of previous uniprocessor designs, with each core having its own private L2 cache. In most server applications, the instruction working set is large and there are sig-

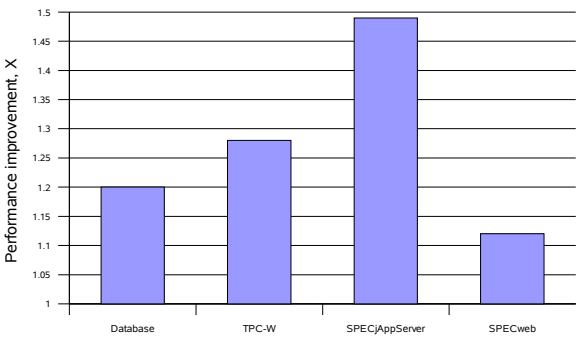

Figure 2. Performance improvements achievable by perfect instruction prefetching (for commercial workloads)

nificant advantages to sharing the L2 cache among the cores on a chip. But a shared L2 cache design requires considerable verification effort to ensure that coherency is maintained correctly. In the second generation of CMT designs, all the cores share the same L2 cache, but the core designs continue to be derived from previous uni-core designs.

Though re-using existing core designs is an attractive option, this approach may not scale well beyond a couple of process generations. Processor designs are already pushing the limits of power dissipation. In order to restrain total power consumption, the power dissipation of each core must be halved in each generation. In the past, supply voltage scaling delivered most of the required power reduction, but indications are that voltage scaling will not be sufficient by itself. Though well-known techniques, such as clock gating and frequency scaling, may be quite effective in the short term, more research is needed to develop low-power high-performance cores for future CMT designs.

In order to maintain the same amount of off-chip bandwidth per core, the total off-chip bandwidth for the processor chip must also double every process generation. This bandwidth increase can be met by increasing the number of pins and/or increasing the bandwidth per pin. However, the maximum number of pins per package is increasing gradually at a rate of 10% per generation [20]. Furthermore, packaging costs per pin are barely decreasing with each new generation and increase significantly with pin count [20]. Thus, the predominant focus has been on increasing the per-pin bandwidth via innovations in the processor chip to DRAM memory interconnect through technologies such as DDR, DDR2 and FBD.

As a result, future CMT processors are likely to be designed from the ground-up to satisfy these power and bandwidth constraints, while delivering ever increasing performance. In this context, it is useful to understand the CMT design space. A strand requires several chip-level resources, including, register file, execution pipeline, floating-point units, branch predictors, L1 I-cache, L1 D-cache, and L2 cache.

In most CMT designs, it is preferable to share the L2 cache, because it localizes coherency traffic between the strands and optimizes inter-strand communication in the chip. However, there are also other design options that should be explored and evaluated with regard to their feasibility and performance potential. For instance, the strands in MAJC share a single L1 data cache, but have separate L1 instruction caches. In Niagara, all strands share a floating-point unit. Furthermore, all strands could share an area-intensive, say perceptron-based, second-level branch predictor for predicting long-latency unresolvable branches (unresolvable branches depend on a load miss and can be resolved only after the load miss returns).

Given the significant area cost associated with high-performance cores, for a fixed area and power budget, the CMP design choice is between a small number of high performance (high frequency, aggressive OoO, large issue width) cores or multiple simple (low frequency, inorder, limited issue width) cores.

For workloads with sufficient TLP, the simpler core solution may deliver superior chip-wide performance at a fraction of the power.

However, for applications with limited TLP, unless speculative parallelism can be exploited, CMT performance will be poor. One possible solution is to support heterogeneous cores, potentially providing multiple simple cores for thread-rich workloads and a single more complex core to provide robust performance for single threaded applications [21].

Another interesting opportunity for CMT processors is support for on-chip hardware accelerators. Hardware accelerators provide increased performance on certain specialized tasks and offload work from the general-purpose processor. Additionally, on-chip hardware accelerators may be an order of magnitude more power efficient than the general-purpose processor and may be significantly more efficient than off-chip accelerators (e.g. eliminating the off-chip traffic required to communicate to an off-chip accelerator). While typically unattractive for traditional processors, due to the high cost and low utilization, because of the high degree of resource sharing associated with CMTs, the cost of an accelerator can be amortized over many strands. While a wide variety of hardware accelerators can be envisaged, emerging trends make support for on-chip network offload engines and cryptographic accelerators extremely compelling.

In the future, there may be opportunities for accelerating other functionality. For instance, IBM has indicated interest in accelerating certain OS functionality and, with the increasing usage of XML formatted data, it may become attractive to provide hardware support for XML parsing and processing.

4 CMT challenges

In a traditional processor design, attention is focused on single thread performance. As a result, microarchitectural innovation has led to increasingly aggressive speculative techniques that often expend significant resources to achieve limited performance improvements. In a single core design, there is limited downside to such a strategy because, during periods of extensive speculation, the resources are otherwise underutilized. Previous research into SMT processors has developed strategies for efficient sharing of key resources such as the issue window, execution pipeline and fetch bandwidth [22]. In a CMT processor, most chip-level resources are shared and further research is needed to identify policies and mechanisms that will maximize overall performance, while not starving any of the strands. In essence, CMT mechanisms must ensure that strands are “good neighbors”. The re-

mainder of this section discusses some of the challenges that arise predominantly due to resource sharing and associated inter-strand interference.

4.1 Prefetching strategies

Speculative prefetching techniques are still important in CMT processors. For instance, the effect of L2 cache stalls due to the instruction stream becomes minimal once the L2 cache size exceeds 2MB on conventional processors and, as a result, instruction prefetching has limited headroom. However, instruction prefetching can result in significant performance benefits for CMT processors, as is illustrated in Figure 2 (throughout this paper the modeled system is a 4-core CMT processor; each core has a private 32KB L1 instruction and data cache and share a unified 2MB L2 cache), because a greater portion of the L2 cache is now occupied by data lines. This is an interesting example of how the CMT design point can reemphasize old problems.

Furthermore, the instruction prefetching strategy must be tailored to the CMT design point. In a single-core design, virtually the entire system is idle on an instruction miss and aggressive prefetching, such as next-four line prefetching, may be very effective. In a CMT processor, a less aggressive prefetching scheme, such as next-line prefetching may be appropriate, depending on the utilization of the memory ports by the other strands.

While incorrect speculation can have harmful side-effects on traditional processors, the potential for problems is heightened on CMT processors. A mis-speculation, such as a wrong prefetch, can delay the use of a resource, such as the memory port, by another strand and may further cause cache pollution by evicting a useful cache line. Hence, accurate speculation becomes a more important design consideration.

4.2 Request prioritization

Hardware scouting or run-ahead execution [23] generates a highly accurate stream of prefetches and pollution is not a major consideration. In a single-strand design, the processor does not issue any demand fetches during scouting and the timeliness of demand fetches is largely unaffected. However, in a CMT processor, when one strand is scouting, the other strands may be issuing demand fetches. In Figure 3, we illustrate that the prefetches generated by hardware scouting significantly delay the demand fetches and potentially impact overall performance.

The L2 cache and the memory controller may give higher priority to demand fetches than to prefetches. Our initial evaluation shows that the displaced prefetches generated by hardware scouting should be delayed, but not dropped. Such request prioritization delivers better overall performance. More sophisticated request prioritization approaches may yield even better performance, especially when cache and memory ports approach saturation levels and speculative requests have varying degrees of accuracy.

4.3 Hot sets & hot banks

The hot set problem occurs in caches when many heavily accessed physical addresses map to the same set, resulting in thrashing and a significant increase in conflict misses. CMT processors often run the same binary on all strands, which leads to better instruction cache performance and more constructive sharing in the L2 cache. However, such workloads may also significantly exacerbate any hot set problems.

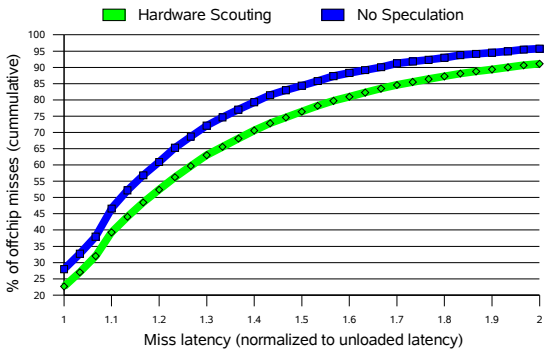

Figure 3. Increased latency of offchip misses resulting from hardware scouting (database workload on a heavily loaded system)

Traditionally, this problem has been addressed by increasing the associativity of the cache. However, beyond a certain level, increasing the associativity also increases access time, preventing the associativity being scaled in line with the number of sharers on CMT systems.

Standard set selection schemes use low-order bits of the physical address to access a set. Set index hashing [24] uses many more bits of the physical address to generate an XOR hash, which is then used to access a set. Such hashing may reduce the prevalence of hot sets. However, the set of address bits that can be used for hashing is often restricted by other considerations. For instance, the L1 cache has to be accessed using only the address bits that are untranslated even when the smallest page size is used, thus limiting the effectiveness of index hashing in L1 caches. Some inclusion properties may need to be maintained between L1 cache and L2 cache banks in order to maintain L1 cache coherence, which may again limit the extent of index hashing used in L2 caches. Thus, there is room for further innovation to solve the hot sets problem, while satisfying the current constraints on L1 and L2 set selection.

In a CMT processor with write-through L1 caches (commonplace, as they make the coherency protocols simpler), all stores access the L2 cache. In order to provide sufficient bandwidth for these stores, as well as L1 load misses and instruction misses, the L2 cache is heavily banked. Bank selection is typically based on low-order bits of the cache line address in order to distribute an unit-stride stream across the banks. However, some banks may be accessed much more frequently than other banks, leading to hot banks and performance problems. Hashing for bank selection is an option, but some of the previous constraints apply.

4.4 Offchip bandwidth

Off-chip bandwidth is another potential performance bottleneck, limiting the number and aggressiveness of the cores in a CMT processor. Figure 4 illustrates how the bandwidth requirements of a core rapidly increase as speculative techniques such as control speculation, hardware scouting and value prediction are utilized.

To an extent, this bottleneck can be mitigated by increasing the amount of on-chip cache, decreasing the offchip miss rate. However, increasing the transistor budget devoted to on-chip caches reduces the area which can be devoted to cores.

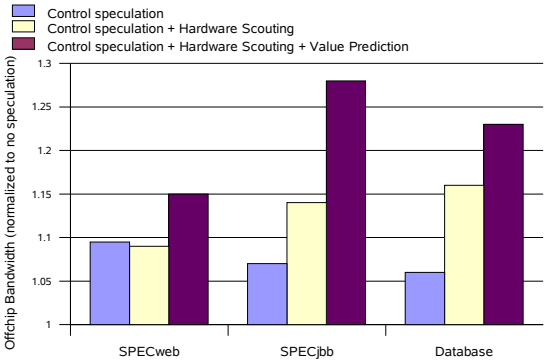

Figure 4. Increase in offchip bandwidth when aggressive speculation is utilized

An alternative strategy is to attempt to do more with the available bandwidth; for instance, by compressing off-chip traffic [25] or exploiting silentness to minimize the bandwidth required to perform writeback operations [26].

Compression of the onchip caches themselves can also improve performance, but the (significant) additional latency that is introduced as a result of the decompression overhead must be carefully balanced against the benefits of the reduced miss rate, favoring adaptive compression strategies [27].

4.5 Compiler optimizations

Compilers can also make a significant difference to application performance on CMT processors. In recent years, aggressively optimizing compilers that perform loop unrolling and aggressive data prefetching have become prevalent. This approach may not be optimal for CMT systems, which exhibit decreased hardware resources per strand. For instance, optimizing for reduced code footprint may make more sense.

5 Conclusion

CMT processors support many hardware strands through efficient sharing of on-chip resources such as pipelines, caches and predictors. CMT processors are a good match for server workloads, which have high levels of TLP and relatively low levels of ILP. The first generation of CMTs were rapidly derived from existing uni-processor designs, whereas the second generation of CMTs support effective sharing of L2 caches. The third generation of CMTs, which have cores specifically designed for CMT systems, are exemplified by Sun's recently announced Niagara processor. Speculative parallelization approaches are important for applications that do not have sufficient TLP. In applications with sufficient TLP, the sharing of resources between strands and the consequent interaction between strands through these shared resources give rise to many design issues. In this paper, we describe some of the open design issues such as hot sets, speculative prefetches, and request prioritization which can benefit from further academic research.

References

[1] K. Olukotun et al., "The Case for a Single-Chip Multiprocessor," in *Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 1996, pp. 2–11.

[2] D. Tullsen, S. Eggers, and H. Levy, "Simultaneous Multithreading: Maximizing On-Chip Parallelism," *Intl. Symp. on Computer Architecture*, pp. 392–403, 1995.

[3] C. Batten, et. al., "Cache Refill/Access Decoupling for Vector Machines," to appear *Intel. Symp. on Microarchitecture*, 2004.

[4] Y. Chou, B. Fahs, and S. Abraham, "Microarchitecture Optimizations for Exploiting Memory-Level Parallelism," in *Intl. Symp. on Computer Architecture*, pp. 76–87, 2004.

[5] S. Gochman et al., "Intel Pentium M Processor: Microarchitecture and Performance", in *Intel Technology Journal*, Vol. 7, No. 2, pp. 22–36, 2003.

[6] V. Krishnan and J. Torrellas, "A Chip-Multiprocessor Architecture with Speculative Multithreading", *IEEE Trans. on Computers*, pp. 866–880, 1999.

[7] L. Hammond et al., "The Stanford Hydra CMP," in *IEEE MICRO Magazine*, Vol. 20, No. 2, pp. 71–84, 2000.

[8] L. A. Barroso et al., "Piranha: a Scalable Architecture Based on Single-Chip Multiprocessing," in *Intl. Symp. on Computer Architecture*, 2000, pp. 165–175.

[9] M. Tremblay et al., "The MAJC Architecture: A Synthesis of Parallelism and Scalability," *IEEE Micro*, Vol. 20, No. 6, pp. 12–25, 2000.

[10] M. Tremblay, "MAJC-5200: A VLIW Convergent MPSOC," in *Microprocessor Forum 1999*, 1999.

[11] S. Kapil, "UltraSPARC Gemini: Dual CPU Processor," in *Hot Chips 15*, `http://www.hotchips.org/archive/`, 2003.

[12] Q. Jacobson, "UltraSPARC IV Processors," in *Microprocessor Forum 2003*, 2003.

[13] P. Kongetira, "A 32-way Multithreaded SPARC Processor," in *Hot Chips 16*, `http://www.hotchips.org/archive/`, 2004.

[14] C. Moore, "POWER4 System Microarchitecture," in *Microprocessor Forum*, 2000.

[15] R. Kalla, B. Sinharoy, and J. Tendler , "IBM POWER5 chip: a dual-core multithreaded processor," in *IEEE Micro*, Vol. 24, No. 2, pp. 40–47, 2004.

[16] Advanced Micro Devices, "AMD Demonstrates Dual Core Leadership," `http://www.amd.com/`, 2004.

[17] T. Maruyama, "SPARC64 VI: Fujitsu's Next Generation Processor," in *Microprocessor Forum 2003*, 2003.

[18] C. McNairy and R. Bhatia, "Montecito - the Next Product in the Itanium Processor Family," in *Hot Chips 16*, `http://www.hotchips.org/archive/`, 2004.

[19] P. Otellini, "Intel Keynote," in *Intel Developer Forum*, 2004.

[20] "International Technology Roadmap for Semiconductors: Executive Summary," `http://public.itrs.net/`, 2003.

[21] R. Kumar, et al., "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction," in *Intl. Symp. on Microarchitecture*, 2003, pp. 81–92.

[22] A. El-Moursy and D. Albonesi, "Front-End Policies for Improved Issue Efficiency in SMT Processors", in *Intl. Symp. on High-Performance Computer Architecture*, 2003, pp. 31–42.

[23] O. Mutlu, J. Stark, C. Wilkerson, and Y. Patt, "Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-Order Processors," in *Intl. Symp. on High-Performance Computer Architecture*, 2003, pp. 129–141.

[24] A. Gonzalez et al., "Eliminating Cache Conflict Misses through XOR-based Placement Functions," in *Intl. Conf. on Supercomputing*, 1997, pp. 76–83.

[25] J. Goodman, D. Burger, and A. Kagi, "Memory Bandwidth Limitations of Future Microprocessors," in *Intl. Symp. on Computer Architecture*, 1996, pp. 78–89.

[26] K. Lepak and M. Lipasti, "Temporally Silent Stores," in *Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, 2002, pp. 30–41.

[27] D. Wood and A. Alameldeen, "Adaptive Cache Compression for High-performance Processors," in *Intl. Symp. on Computer Architecture*, 2004, pp. 212–223.