Invited Paper:

Toward Energy-Efficient High-Performance Organizations of the Memory Hierarchy in Chip-Multiprocessor Architectures

Francisco J. Villa, **Manuel E. Acacio** and **José M. García**
Dept. Ingeniería y Tecnología de Computadores
Universidad de Murcia, SPAIN
E-mail: {fjvilla, meacacio, jmgarcia}@ditec.um.es

ABSTRACT

Chip-multiprocessor systems or CMPs have emerged as a high-performance organization for the increasing number of transistors available on a chip, and are projected to dominate the market of server and desktop computers. CMPs require innovative designs of on-chip memory hierarchies, especially designed to address the problems that arise in this novel kind of architecture: higher memory bandwidth demand from more processing cores and the increasing latency of off-chip cache misses. Moreover, the energy consumption topic is even more pressing than in traditional multiprocessors, as the CMPs are commonly used in embedded systems. This paper presents a survey of some of the proposals that have recently appeared facing these topics.

Keywords: Chip-multiprocessors, Memory Hierarchy Organization, Cache Miss Latency, Cache Storage Use, Energy-Aware Techniques

1 INTRODUCTION

As integration scales grows, the number of transistors available on a die is increasingly becoming larger, and billion transistor chips are possible nowadays [6]. The role of computer designers is to translate all this raw potential into increased computational power. For this, efficient architectures must be designed. One of the approaches recently proposed in the literature to make efficient use of this huge number of transistors is single chip-multiprocessors [3, 10]. A CMP integrates several processor cores onto a chip, as well as other resources such as the cache hierarchy and the interconnection network. As a result, the traditional advantages of parallel architectures are kept, but some of their drawbacks, such as wire delays or network latencies, are minimized.

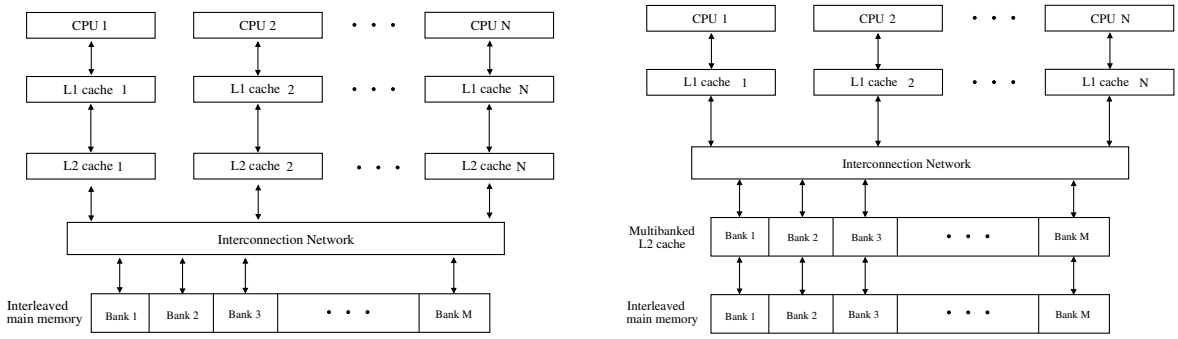
The viability and importance of chip multiprocessors is further supported by a number of recently announced commercial CMP designs [16, 21, 22]. However, nowadays the best organization of the components in this kind of architecture has not been defined, and there is still much work to be done in order to improve the performance and maximize the utilization of the resources in a CMP.

One of the topics that more efforts is deserving by the researchers is the organization of the cache hierarchy [11, 31]. The combination of the high bandwidth demands from the processing cores, together with the high cost of cache misses, makes this issue especially critical for future chip-multiprocessors.

There are two major alternatives for organizing the last level of the cache hierarchy in a CMP [29] (from now on, we suppose that the L2 cache is the last level of on-chip cache): *private* or *shared* L2 cache (see Figure 1). In the private L2 cache organization (Figure 1(a)), the total cache storage is partitioned between the processor cores, and misses from the L1 cache go directly to the private L2 cache without crossing the shared interconnection network. In the shared L2 cache organization (Figure 1(b)), the L2 cache storage is shared by all the processors and cache coherence is maintained at the L1 cache level.

Each approach has different pros and cons. In the private cache organization, most of the L1 cache misses can be handled by the L2 cache, so the number of remote on-chip L2 cache accesses is reduced and it is not necessary to cross the interconnection network, which reduces the miss latency. However, private L2 caches make inefficient use of the total cache storage space, as multiple copies of the same line can reside in different private L2 caches. Another issue regarding the private L2 cache organization is the load balance problem that can arise due to the static assignment of L2 cache space to processor cores.

On the other hand, a shared L2 cache organization can reduce the collective number of L2 misses, as there are not replicated copies of the same line. Besides, the load balancing problem can be solved with a nonuniform distribution of the cache space between the cores. The main drawbacks of this organization are the higher hit latency, as the interconnection network must be used to access remote L2 cache banks, and the possible interference between cores, as each one can evict the blocks that other is using.



(a) Private L2 cache.

(b) Shared L2 cache.

Figure 1. Private vs. shared L2 cache organizations.

Several proposals have recently appeared in the literature that attempt to achieve the benefits of both basic organizations. As we can see, the design of the cache hierarchy in a CMP is a very complex problem in which all the variables previously mentioned must be taken into account.

Another interesting topic to keep in mind when designing a CMP architecture is the thermal/energy consumption. As processors continue to increase in speed, energy consumption and heat dissipation have become key design variables, especially if we take into account that chip-multiprocessors architectures are suited for the embedded computing domain, where the energy consumption restrictions are more pronounced. Increased power consumption and heat dissipation typically leads to higher costs for thermal packaging, fans, electricity, and even air conditioning, so it is imperative to design energy-aware chip-multiprocessor systems.

In this paper, we present a revision of the alternatives that have been recently proposed addressing these two important issues in CMP architectures: cache hierarchy organization and energy consumption. First, in Section 2 we depict several proposals aimed at reducing the cache miss latency. In Section 3 we describe some cache organizations aimed at optimizing the use of cache storage. Then, in Section 4, some techniques facing the power consumption problem are presented. Finally, Section 5 concludes the paper.

2 REDUCING CACHE MISS LATENCY

As we have stated in Section 1, private L2 cache organizations suffer from lower L1 cache miss latencies than shared L2 cache architectures at the expense of poor cache storage utilization. In this section, we describe some proposals aimed at approximating the latencies of private caches while maintaining the advantages of shared organizations.

Several authors propose a hybrid cache scheme which tries to retain the advantages of a shared L2 cache but minimizing the miss latency. In [37], the authors present a new cache management policy called *victim replication*. In their proposal, the L2 cache space is divided into several slices which can be managed as private or shared. The basis of this mechanism consists in allowing multiple copies of a cache line in different slices. In this way, each processor has a dynamically formed private L2 cache.

In Nonuniform Cache Access (NUCA) designs, the problem of on-chip wire delays is addressed. This problem makes large shared L2 caches with an unique hit latency a bad design choice [1]. Several on-chip cache designs exploiting the NUCA concept are evaluated in [18, 19]. The S-NUCA-1 design features a two-level cache hierarchy in which the inclusion property is not maintained. The mapping of data into banks is performed statically and each bank uses a pipelined transmission channel to service requests. S-NUCA-2 improves the previous design by using a two-dimensional switched network, as the area requirements of using a dedicated transmission channel becomes prohibitive when the number of L2 cache banks increases. A third design named D-NUCA includes a mechanism of block promotion in order to exploit the lower hit latency of nearer cache banks. In this architecture, blocks are migrated to faster banks (the nearest banks to the consumer) in order to reduce L2 hit latency.

The D-NUCA concept is applied to CMP architectures in [12], where the authors evaluate different sharing degrees in order to find the optimum compromise between cache space utilization and hit latency. Two migration policies are investigated: D-NUCA 1D, which allows migration only in the vertical dimension, and D-NUCA

2D, in which migration can happen in the horizontal dimension as well. In order to prevent the situation in which a block is migrated from one cache bank to another because two processors are accessing it, the authors propose a two-bit saturating counter which allows the migration only if the counter corresponding to that direction is saturated.

The proposals contained in [34] refer to a hybrid CMP/SMT architecture with two levels of private cache and a third level of on-chip shared cache. As the L2 cache access latency is much lower than the L3 cache one, they consider the case in which write backs from an L2 cache can be placed in a neighbouring cache instead of the further L3 cache. They use a small table to keep track of those lines with a high reuse potential. This table annotates the lines that were written back and later reused by another L2 cache.

In [4] the authors evaluate three well-known uniprocessors techniques for reducing L2 cache latency. The first technique is hardware-directed strided prefetching [14]. They use L1 and L2 prefetching and find that this technique can hide L2 cache latency in great extent. The second technique evaluated is the D-NUCA block migration policy previously commented. They observe that without an intelligent block migration algorithm, as the one proposed in [12], shared blocks tend to migrate to the central cache banks, which are equally distant from all the processors. Finally, transmission line caches (in which on-chip transmission lines are used in order to provide low access latency to all the banks) are evaluated; the technique exhibits similar results to those found in uniprocessor architectures, but the contention in the dedicated lines becomes an important fraction of the overall L2 hit latency. The authors evaluate a hybrid scheme which combines the three techniques and propose using transmission lines only to access the central banks where the cache lines are concentrated due to block migration.

Another proposal related to the on-chip communication mechanisms can be found in [5]. In this paper, the authors optimize the producer-consumer coherence pattern, in which the copy of the line in the reader cache is invalidated on every write and subsequently read again. They propose an implementation in which a single copy of the block is maintained for the lines that exhibit this behavior. In their implementation, the copy is stored close to the reader, as they observe that in their evaluation environment (composed of commercial workloads) each write is read more than once by each reader. They add a communication state to the coherence protocol in which the writer can write the block and the readers can read it without incurring in a coherence miss.

Synchronization using a shared variable is a well-known problem in shared-memory multiprocessors [24]. This problem is addressed in the context of a multiprocessor-on-a-chip in [36]. The authors propose a mechanism named *Tagged Shared Variable Memory* (TSVM) which stores the shared variables used in synchronization in a dedicated cache. The TSVM is divided into two structures: the TSVM cache (TC), which works as a L1 cache for synchronization variables, and a conventional memory. Thus, the L1 cache can be viewed as a memory with a dedicated portion for synchronization variables (the TC) and a *General Variable Cache* (GVC) that caches the remaining variables. As the hit latency for synchronization operations is reduced, the performance of the applications evaluated is boosted notably.

3 OPTIMIZING THE USE OF CACHE STORAGE

While the proposals described in the previous section are aimed at reducing the cache access latency, some recently proposed mechanisms try to optimize the use of cache capacity.

Liu *et al.* investigate different organizations for the L2 cache in [29], addressing the tradeoffs between private and shared L2 cache organizations. They propose a hybrid organization for the L2 cache called *Shared-Processor Based Split*. In this organization, accesses to the cache are based on the CPU cores rather than on the memory address (as in typical shared L2 caches), which allows better balance of the load between processors. The total L2 cache space is divided into several chunks which are assigned dynamically to the processors. In this way, the Shared-Processor Based Split L2 cache provides a flexible way of configuring the L2 cache based on workload characteristics.

Block replication is the basis for some of the proposals contained in [5]. In order to prevent the waste of on-chip capacity caused by uncontrolled replication, they avoid this replication when a copy of the shared line exists in the private cache of another processor. As the copy is on-chip, the latency penalty is small and it does not overcome the benefits of the improved use of cache storage. Another mechanism proposed in the same paper controls the case in which a core needs more capacity than the provided by its private cache. With the *capacity stealing* technique, the less-frequently-used blocks of the core which needs more cache capacity can be placed in the caches of the closer cores with less capacity demands. This strategy reduces the higher miss rates typical in private cache organizations.

The authors of [34] propose a small history table in order to perform smart write backs in the context of a three-level cache hierarchy with two levels of private cache. On a replacement of a clean line, the cache controller consults this table in order to know if there is another copy of the line in an L2 cache or in the L3 cache, so that it is possible to make a more informed decision about writting back the line or not. This table is associated with each L2 cache in the system and is organized as a cache tag array.

Iyer proposed in [13] a priority-based cache management mechanism to allocate cache resources by priority. The paper discusses several options for priority classification, assignment and enforcement, placing special emphasis on the last topic. To enforce priorities in the framework, three possible schemes are discussed: static/dynamic partitioning (higher priority applications have more ways in cache sets than lower priority ones), selective cache allocation (allow/disallow allocation based on the current cache space used and the priority given to each kind of traffic) and heterogeneous cache regions (where heterogeneous cache structures with different set associative, for example, are mapped to memory access streams with different priorities).

Settle *et al.* study the effects of thread interference in *simultaneous multithreading* CMP architectures [33]. Interference occurs in the cache system when data belonging to one thread is evicted by a cache line from another thread. To solve this problem, they also use cache partition techniques, and the space assigned to each thread is reassigned dynamically based on the degree of global data reuse of lines previously evicted and the percentage of cache storage already allocated to the thread. In this way, threads that exhibit low reuse due to the lack of cache storage can compete with threads showing high reuse because they own most of the lines.

Another work related to CMP/SMT architectures is [35]. This work presents a cache partitioning scheme that dynamically allocates cache space using a set of counters. These counters are used to estimate the changes in process miss rate when allocating/deallocating cache storage to that processor. The replacement unit keeps track of the number of cache blocks belong to each active process, and allocates a new cache block to a process only if its current allocation is below its limit.

Kim *et al.* presents a detailed study of fairness in cache sharing between threads in [20], where fairness measures how the threads are slowed down due to cache sharing. They propose five cache fairness metrics and static/dynamic cache partitioning algorithms which use these metrics. The dynamic algorithm does not restrict the cache replacement algorithm to LRU, as some of the proposals described previously require. Finally, the paper correlates fairness and throughput and shows that optimizing fairness usually increases throughput.

4 ENERGY-AWARE TECHNIQUES IN CMPS

While the proposals summarized in the previous sections are aimed at improving the performance of the memory subsystem in chip-multiprocessors, another important topic regarding this kind of architecture is the energy consumption, as we have discussed in Section 1. This section covers this issue and describes some energy-aware proposals.

Some of these proposals are based on the dynamic voltage/frequency scaling. A first work by Li and Martínez [27] develops an analytical model in which the variables are efficiency, application granularity and voltage/frequency scale. The work shows that tuning the levels of voltage/frequency correctly, great energy savings can be obtained. However, the choice is dependent on the application and the process technology, which complicates this voltage/frequency scaling.

The previous work is extended in [26], where the authors propose a mechanism which optimizes dynamically the energy consumption of a parallel application running on a CMP composed of 16 processor cores under certain performance restrictions. They present a set of heuristics applied to the number of active processor cores and the dynamic voltage/frequency scaling, which are typically the two most important dimensions of the energy-aware design space.

The proposal contained in [15] follows a compiler-based approach. The compiler takes advantage of the heterogeneous parallel execution of the workloads in different processors and assigns different voltage/frequency pairs to different processors if the energy consumption is reduced and, at the same time, the increasing of the execution time is not significant.

Another approach followed in some works to reduce the energy waste relies on the use of heterogeneous multi-core architectures. In [23] several single-ISA heterogeneous multi-core designs are proposed and evaluated in the context of multiprogrammed workloads. The different cores represent some points in the performance/energy design space; during the execution of an application, the system software chooses dynamically the more appropriate core in order to meet with the specific requirements of energy and performance of the application.

The goal pursued in [2] is minimizing the execution time of multithreaded programs containing parallel and sequential phases in their execution, while maintaining the energy consumption within a limited interval. In this case, the *Energy per Instruction* (EPI) parameter of the formula $Power = EPI * InstructionsPerSecond(IPS)$ is modified according to the amount of available parallelism in an asymmetric multiprocessor with four cores.

Another evaluation of a CMP environment limited by the energy can be found in [9], where the authors evaluate the compromise between latency performance and throughput performance. Their proposal is also based on varying the amount of energy wasted to process the instructions taking into account the level of parallelism available in the software. To accomplish this, they use a combination of well-known energy-aware techniques: voltage/frequency scaling, asymmetric cores, heterogeneous cores and speculation control.

The problem of thread synchronization using barriers is addressed in [30]. By using a high-level synchronization constructor, the proposed technique tracks the idle intervals in which a processor is waiting for the other processors to reach the same point in the program. Using this information, the technique modules the frequency of the different processors in order to remove the idle intervals and then reducing the energy consumption.

The compromise between the issue width of processor cores and the number of cores on a chip is analyzed in [8]. All the designs evaluated feature comparable sizes and the comparison is performed with respect to both performance and energy waste. The results show that it is possible to achieve the same performance with narrower cores and with lower energy consumption.

A recent work [28] shows that the thermal restrictions dominates over other physical restrictions in CMP architectures. As an important conclusion, for aggressive cooling solutions reducing power density is at least as important as reducing total power, while for low-cost cooling solutions they suggest that reducing total power is more important.

Finally, there are a couple of works dealing with CMP/SMT implementations and the energy efficiency of these two families of architectures. A work measuring the potential for techniques aimed at reducing the energy consumption in these architectures appears in [7], where the authors find that both architectures present prominent temperature gradients but, at the same time, they show a great potential for temperature-aware enhancements to mitigate the problem.

Kaxiras *et al.* evaluate the power consumption of a SMT and a CMP DSP for mobile phone workloads [17]. They find that the SMT VLIW DSP uses up to 40% less power than the CMP DSP in their target environment. The results are different in an environment composed of out-of-order processor cores, like the one evaluated in [32]. In this case, the results show that the CMP alternative is more efficient in terms of energy consumption but requires more area to be implemented, so the authors propose a hybrid architecture where a CMP integrates several SMT cores. Taking the number of pipeline stages and the pipeline width as metrics of the core complexity, Lee and Brooks evaluate the energy/performance tradeoff of several SMT and CMP architectures of varying complexity [25]. Their results suggest that SMT architectures enable efficient increases in pipeline dimensions and core complexity, while reducing pipeline dimensions in CMP cores is inefficient.

5 CONCLUSIONS

Chip-multiprocessor systems or CMPs have emerged as a high-performance organization for the increasing number of transistors available on a chip, and are projected to dominate the market of server and desktop computers. However, the best organization of the components in this kind of architecture has not been defined yet, and there is still much work to be done in order to improve the performance and maximize the utilization of the resources in a CMP.

This paper surveys some of the proposals that have recently appeared focusing on two of the more critical issues when designing a CMP architecture: the organization of the cache hierarchy and the use of techniques aimed at reducing the energy consumption. In the organization of the cache hierarchy we have shown possible ways of reducing the latency of the cache misses and a couple of proposals that attempts to improve the use of the cache storage. Regarding the energy consumption, several works show the potential of future energy-aware techniques applied to CMPs and some optimizations in the domains of voltage/frequency scaling, thread synchronization, heterogeneous multi-core architectures, issue width and hybrid CMP/SMT architectures have been described.

ACKNOWLEDGMENTS

This work has been supported by the Spanish Ministry of Educación y Ciencia and the European Union (Feder Funds) under grant TIC2003-08154-C06-03.

References

[1] V. Agarwal, M.S. Hrishikesh, S. W. Keckler, and D. Burger. "Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures". In *Proc. of 27th Int'l Symp. on Computer Architecture*, pages 248–259, June 2000.

[2] M. Annavaram, E. Grochowski, and J. Shen. "Mitigating Amdahl's Law Through EPI Throttling". In *Proc. of 32th Int'l Symp. on Computer Architecture*, pages 298–309, June 2005.

[3] L. A. Barroso, K. Gharachorloo, R. McNamara, A. Nowatzyk, S. Qadeer, B. Sano, S. Smith, R. Stets, and B. Verghese. "Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing". In *Proc. of 27th Int'l Symp. on Computer Architecture*, pages 282–293, June 2000.

[4] B. Beckmann and D. Wood. "Managing Wire Delay in Large Chip-Multiprocessor Caches". In *Proc. of 37th Int'l Symp. on Microarchitecture*, pages 319–330, December 2004.

[5] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. "Optimizing Replication, Communication, and Capacity Allocation in CMPs". In *Proc. of 32th Int'l Symp. on Computer Architecture*, pages 357–368, May 2005.

[6] Intel Corporation. *"Dual-Core Update to the Intel Itanium 2 Processor. Reference Manual"*, January 2006.

[7] J. Donald and M. Martonosi. "Temperature-Aware Design Issues for SMT and CMP Architectures". In *Proc. of 2004 Workshop on Complexity Effective Design*, June 2004.

[8] M. Ekman and P. Stenstrom. "Performance and Power Impact of Issue-width in Chip-Multiprocessor Cores". In *Proc. of Int'l Conf. on Parallel Processing*, pages 359–368, October 2003.

[9] E. Grochowski, R. Ronen, J. Shen, and H. Wang. "Best of Both Latency and Throughput". In *Proc. of Int'l Conf. on Computer Design*, pages 236–243, October 2004.

[10] L. Hammond, B. A. Hubbert, M. Siu, M. K. Prabhu, M. Chen, and K. Olukotun. "The Stanford Hydra CMP". *IEEE Micro*, 20(2):71–84, March 2000.

[11] J. Huh, D. Burger, and S. W. Keckler. "Exploring the Design Space of Future CMPs". In *Proc. of 2001 Int'l Conf. on Parallel Architectures and Compilation Techniques*, pages 199–210, September 2001.

[12] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler. "A NUCA Substrate for Flexible CMP Cache Sharing". In *Proc. of 10th Int'l Conf. on Supercomputing*, pages 31–40, June 2005.

[13] R. Iyer. "CQoS: A Framework for Enabling QoS in Shared Caches of CMP Platforms". In *Proc. of Int'l Conf. on Supercomputing*, pages 257–266, June 2004.

[14] N. P. Jouppi. "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers". In *Proc. of 17th Int'l Symp. on Computer Architecture*, pages 364–373, May 1990.

[15] I. Kadayif, M. Kandemir, and I. Kolcu. "Exploiting Processor Workload Heterogeneity for Reducing Energy Consumption in Chip Multiprocessors". In *Proc. of Design, Automation and Test in Europe Conference and Exhibition*, pages 882–887, February 2004.

[16] R. Kalla, B. Sinharoy, and J. M. Tendler. "IBM Power5 Chip: A Dual-Core Multithreaded Processor". *IEEE Micro*, 24(2):40–47, March-April 2004.

[17] S. Kaxiras, G. Narlikar, A. D. Berenbaum, and Z. Hu. "Comparing Power Consumption of an SMT and a CMP DSP for Mobile Phone Workloads". In *Proc. of Int'l Conf. on Compilers, Architecture and Synthesis for Embedded Systems*, pages 211–220, November 2001.

[18] C. Kim, D. Burger, and S. W. Keckler. "An Adaptive, Non-Uniform Cache Structure for Wire-Dominated On-Chip Caches". In *Proc. of 10th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pages 211–222, October 2002.

[19] C. Kim, D. Burger, and S. W. Keckler. "Nonuniform Cache Architectures for Wire-Delay Dominated On-Chip Caches". *IEEE Micro*, 23(6):99–107, November/December 2003.

[20] S. Kim, D. Chandra, and Y. Solihin. "Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture". In *Proc. of 10th Int'l Conf. on Parallel Architecture and Compilation Techniques*, pages 111–122, September 2004.

[21] K. Krewell. "UltraSPARC IV Mirrors Predecessor". Micro. Report, pp. 1-3, November 2003.

[22] K. Krewell. "Sun's Niagara pours on the cores". *Microprocessor Report*, 18(9):11–13, September 2004.

[23] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen. "Single-ISA Heterogeneous Multi-core Architectures: The Potential for Processor Power Reduction". In *Proc. of Int'l Symp. on Microarchitecture*, pages 81–92, December 2003.

[24] S. Kumar, D. Jiang, R. Chandra, and J. P. Singh. "Evaluating Synchronization on Shared Address Space Multiprocessors: Methodology and Performance". In *Proc. of Int'l Conf. on Measurement and Modeling of Computer Systems*, pages 23–34, May 1999.

[25] B. C. Lee and D. Brooks. "Effects of Pipeline Complexity on SMT/CMP Power-Performance Efficiency". In *Proc. of 2005 Workshop on Complexity Effective Design*, June 2005.

[26] J. Li and J. F.Martínez. "Dynamic Power-Performance Adaptation of Parallel Computation on Chip Multiprocessors". In *Proc. of 12th Int'l Symp. on High-Performance Computer Architecture*, pages 77–87, February 2006.

[27] J. Li and J. F. Martínez. "Power-Performance Implications of Thread-level Parallelism in Chip Multiprocessors". In *Proc. of Int'l Symp. on Performance Analysis of Systems and Software*, pages 124–134, Match 2005.

[28] Y. Li, B. Lee, D. Brooks, Z. Hu, and K. Skadron. "CMP Design Space Exploration Subject to Physical Constraints". In *Proc. of 12th Int'l Symp. on High Performance Computer Architecture*, pages 15–26, February 2006.

[29] C. Liu, A. Sivasubramaniam, and M. Kandemir. "Organizing the Last Line of Defense before Hitting the Memory Wall for CMPs". In *Proc. of 10th Int'l Symp. on High Performance Computer Architecture*, pages 176–185, February 2004.

[30] C. Liu, A. Sivasubramaniam, M. Kandemir, and M. J. Irwin. "Exploiting Barriers to Optimize Power Consumption of CMPs". In *Proc. of 19th Int'l Parallel and Distributed Processing Symp.*, pages 5a–5b, April 2005.

[31] B. A. Nayfeh, L. Hammond, and K. Olukotun. "Evaluation of Design Alternatives for a Multiprocessor Microprocessor". In *Proc. of 23rd Int'l Symp. on Computer Architecture*, pages 66–77, June 1996.

[32] R. Sasanka, S. V. Adve, Y. Chen, and E. Debes. "The Energy Efficiency of CMP vs. SMT for Multimedia Workloads". In *Proc. of Int'l Conf. on Supercomputing*, pages 196–206, June 2004.

[33] A. Settle, D. Connors, E. Gibert, and A. Gonzalez. "A Dynamically Reconfigurable Cache for Multithreaded Processors". *Journal of Embedded Computing: Special Issue on Single-Chip Multi-core Architectures*, December 2005.

[34] E. Speight, H. Shafi, L. Zhang, and R. Rajamony. "Adaptive Mechanisms and Policies for Managing Cache Hierarchies in Chip Multiprocessors". In *Proc. of 32th Int'l Symp. on Computer Architecture*, pages 346–356, May 2005.

[35] G. E. Suh, L. Rudolph, and S. Devadas. "Dynamic Cache Partitioning for CMP/SMT Systems". *Journal of Supercomputing*, 28(1):7–26, April 2004.

[36] A. Yamawaki and M. Iwane. "Organization of Shared Memory with Synchronization for Multiprocessor-on-a-chip". In *Proc. of 9th Int'l Conf. on Parallel and Distributed Systems*, pages 83–90, December 2002.

[37] M. Zhang and K. Asanovic. "Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors". In *Proc. of 32nd Int'l Symp. on Computer Architecture*, pages 336–345, June 2005.