Ring Data Location Prediction Scheme for Non-Uniform Cache Architectures

Sayaka Akioka
Waseda University
akioka@muraoka.info.waseda.ac.jp

Feihui Li
NVIDIA
fli@nvidia.com

Konrad Malkowski
The Pennsylvania State University
malkowsk@cse.psu.edu

Padma Raghavan
The Pennsylvania State University
raghavan@cse.psu.edu

Mahmut Kandemir
The Pennsylvania State University
kandemir@cse.psu.edu

Mary Jane Irwin
The Pennsylvania State University
mji@cse.psu.edu

***Abstract*— Increases in cache capacity are accompanied by growing wire delays due to technology scaling. Non-Uniform Cache Architecture (NUCA) is one of proposed solutions to reducing the average access latency in such cache designs. While most of the prior NUCA work focuses on data placement, data replacement, and migration related issues, this paper studies the problem of data search (access) in NUCA. In our architecture we arrange sets of banks with equal access latency into rings. Our Last Access Based (LAB) prediction scheme predicts the ring that is expected to contain the required data and checks the banks in that ring first for the data block sought. We compare our scheme to two alternate approaches: searching all rings in parallel, and searching rings sequentially. We show that our LAB ring prediction scheme reduces L2 energy significantly over the sequential and parallel schemes, while maintaining similar performance. Our LAB scheme reduces energy consumption by 15.9% relative to the sequential lookup scheme, and 53.8% relative to the parallel lookup scheme.**

I. Introduction

The on-chip L2 and L3 cache capacities continue to increase to accommodate growing data-set sizes and pressures coming from multi-threaded applications. As hardware manufacturers move toward smaller process technologies, the global wire delays increase, resulting in high access latencies for large uniform access cache structures. One way of coping with these increasing latencies is to divide the cache into banks that exhibit nonuniform access latencies. A cache architecture with different access latencies to banks is called NUCA [1], [2].

In a NUCA-based system, the L2 space is organized as a set of banks connected to each other using an on-chip network [3], [4], [5], [6], [7]. Each of these banks can be accessed independently, with access latency dependent on the physical distance of the bank from the CPU. We use the term "ring" to denote a set of banks that exhibit the same access latency based on the Manhattan distance. Figure I shows the logical view of our "ring" based NUCA system.

In this paper, we focus on the data access policy and aim to reduce the average access latency and energy consumption. The two base access schemes are sequential and parallel search, and are similar to the access schemes presented by Kim et al. [1]. In the sequential access scheme, on a data request, the lowest latency ring is queried first. On a miss in that ring, the search expands to outer rings, one at a time, until the requested data is found or an L2 miss is determined. When data is placed in rings near the CPU, this method is power efficient. However, when data is placed in outer rings, this method incurs a significant performance penalty. In the parallel access scheme all rings are queried at the same time, resulting in shorter average access time and a significantly higher energy cost than that of the sequential search. We propose a Last Access Based (LAB) scheme *ring prediction* that strikes a balance between the parallel and sequential search. In our scheme we predict the ring of the bank that contains the data sought, based on past access history, and check all the banks in that ring first. With high *ring locality of data accesses,* we can expect good performance and power behavior from this approach.

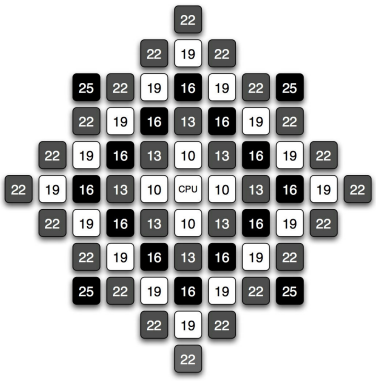


Fig. 1. An example of NUCA topology with six rings and different access latencies.

Our main contributions can be explained as follows:

We propose the Last Access Based ring prediction scheme in which the next access is predicted to hit in the ring that satisfied the current request. Our experimental evaluation of this prediction scheme using the benchmarks from the

978-1-4244-2658-4/08/$25.00 ©2008 IEEE

TABLE I

DEFAULT SYSTEM CONFIGURATION PARAMETERS.

Processor	single-issue, in-order (4GHz)
L1 cache	split I/D, each 32KB, 4-way, 32B line, 5-cycle latency, write-back
L2 cache	unified, 2MB, 8-way, 64B line, 7-cycle per bank access latency
Number of banks	64
Number of rings	6
Per hop latency of network	3 cycles
Data placement policy	banks divided into bank-slices, static placement in slice based on data address, random placement within the slice
Data migration policy	no migration (default) migrate/swap to closer bank (with migration)
Data replacement policy	off-chip (default) swap with the migrated data (with migration)

Spec2000 suite [8] shows that this LAB ring prediction is successful in most of the benchmarks. Specifically, this prediction scheme leads, on average, to 54.8% saving on energy compared to searching all the rings in parallel, with a performance similar to sequential search.

The reminder of this paper is organized as follows. Section II describes our experimental setup. Section III gives the description of two previously proposed simple data access schemes for NUCA. Section IV-A explains the details of our LAB ring prediction scheme and presents an experimental evaluation of it. Section V discusses related work on NUCA and Section VI concludes the paper by summarizing our main observations and presenting future research goals.

II. EXPERIMENTAL SETUP

We use the SIMICS tool set [9] to implement and evaluate our data access policies. We simulate a single-issue, in-order processor running at 4GHz. We calculate the energy values using CACTI [10] and work by Chen et al.[11]. The default system configuration is given in Table I. The lowest access latency to our ring based NUCA L2 cache, *including both network latency and bank access latency,* is 10 cycles and the highest latency is 25 cycles. Our simulations assume that there is no network contention inside of the cache, to simplify and accelerate the simulation process.

To evaluate the performance and energy consumption of LAB scheme for various data access patterns we use the SPEC 2000 benchmark suite. We fast-forward each benchmark to the end of the initialization part, and simulate a 4 billion instruction window. Table II presents the L1 miss rates of our benchmarks, collected using the default values of our simulation parameters listed in Table I.

TABLE II

L1 MISS RATES FOR THE SPEC 2000 BENCHMARKS.

Benchmarks	L1 miss rates
wupwise	1.79%
swim	1.36%
mgrid	1.09%
applu	2.12%
mesa	1.6%
art	14.37%
equake	2.13%
ammp	34.15%
apsi	0.71%
gzip	3.83%
vpr	3.51%
mcf	1.83%
crafty	3.00%
parser	2.99%
gap	0.89%
vortex	2.44%
twolf	7.3%

III. BASE DATA ACCESS SCHEMES

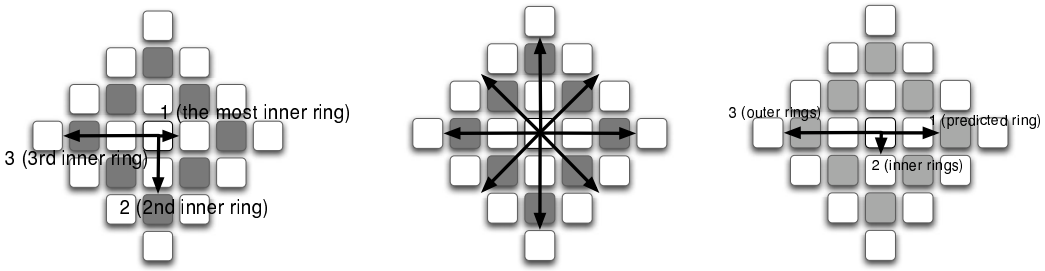
In this section, we discuss the sequential and parallel data search schemes. These two base search schemes do not use any ring prediction. The first of these, called the *sequential scheme* in this paper, accesses the rings one by one, starting with the lowest latency ring, until the requested data is found or a miss is identified (see Figure 2(a)). The second one, the *parallel scheme,* accesses all the rings in parallel, and is illustrated in Figure 2(b). Note that both sequential and parallel schemes were discussed by Kim et al. [1].

There is an trade-off between performance and energy consumption between these two schemes. When no migration scheme is present, the sequential access scheme on average results in degraded performance, when compared to the parallel search scheme. However, because the banks/rings are accessed one a time in the sequential scheme, it usually consumes less energy than the parallel scheme. Additionally, the sequential search scheme places less strain on the on-chip network connecting the banks and cores.

IV. LAST ACCESS BASED (LAB) RING PREDICTOR

A. Description

Our proposed predictor scheme uses the last ring that satisfied a data request to predict the next ring that will be accessed. That is, it predicts that the ring that contains the next data to be accessed is the one that supplied the current data. If the prediction is correct, the data is accessed in one ring access request. If, on the other hand, the prediction is not correct, our approach checks the rings that are enclosed by the predicted ring, starting with the closest ring to the predicted one and ending with the ring that is closest to the CPU. If the requested data is still not found, the remaining rings (i.e., the ones that enclose the originally predicted ring) are accessed sequentially, ending with the outermost ring. If the data is not in one of those rings either, this indicates an L2 miss and the data is fetched from the off-chip memory.



(a) sequential (b) parallel (c) ring prediction

Fig. 2. Different data search (access) schemes in NUCA.

At high level, as shown in Figure 2(c), our proposed ring prediction based scheme, takes at most three steps to locate data (or identify an L2 miss). These steps are: an access to the predicted ring; accesses to the inner rings (sequentially); and accesses to the outer rings (sequentially). When prediction accuracy is high, our ring prediction should have comparable or better performance than the sequential search scheme. In case when the data location is miss-predicted, and data is located in the inner banks, the sequential scheme may be more power efficient. However, if the data is located in middle or outer rings, our approach should consume less energy. Likewise, when the prediction accuracies are high, our LAB scheme should be more power efficient than the parallel scheme. Also, with high prediction accuracies, the performance of LAB should be close to that of the parallel scheme.

We want to emphasize that our prediction based scheme can be used with or without data migration. In fact, our approach is orthogonal to the selection of specific data placement, replacement and migration policies. However, depending on whether data migration is employed or not, the benefits of our approach may vary, since migration brings frequently used data closer, to the rings near the CPU. In our experimental evaluation, we quantify the benefits of our approach with and without data migration within the L2 space. However, in our default configuration, data migration is not used.

B. Evaluation

In this section, we compare our LAB predictor scheme with sequential and parallel search schemes. Prediction accuracy of LAB for our benchmarks is on average 57.7% for floating-point benchmarks, and 43.8% for integer benchmarks, as shown in Figure 3. When our search scheme miss-predicts the ring that should contain the data, 42.7% of the time on average, the data is located in inner rings, and 57.3% of the time in the outer rings. The performances of these three schemes (sequential, parallel and LAB predictor) are compared in Figure 4. While the execution cycles obtained for all three are similar, as expected, the prediction based scheme ranks between the sequential and parallel search schemes (it is 4.1% worse than the parallel search scheme and 0.7% better than the sequential search scheme, on average).

Note that *the energy consumption values include the energies spent during both the network access and the bank access*. The average energy consumption per L2 hit under the three search schemes is plotted in Figure 5. Our prediction based approach performs better than the alternatives, with the average energy savings of 15.9% over the sequential search and 53.8% savings over the parallel search scheme. Overall our LAB search scheme results in energy savings, while offering comparable performance. However, for benchmarks which exhibit low prediction accuracy such as gzip and vpr, our scheme consumes more energy than the sequential search (as can be seen from Figure 5).

We now evaluate sensitivity of our prediction scheme to the bank size (when cache size remains constant) and effects of data migration. For our sensitivity analysis, we focus on four representative benchmarks (two integer and two floating-point). The omitted SPEC benchmarks had similar trends to those presented here. We study the effects of varying the number and size of banks in L2 while maintaining a constant L2 size. Recall from Table I, the default that the number of banks used in the experiments so far was 64 (i.e., 32KB bank size). Impacts of varying the cache bank size on the execution time and average energy consumption per access are shown in Figures 6 and 7, respectively. For each benchmark, results are normalized with respect to the 16KB bank size configuration. We observe that 32KB banks generate the best execution cycle results, and 16KB banks result in the worst cycles. On the other hand, 64KB banks provide most energy savings, and 32KB banks are the second best regarding the energy savings.

The next issue we study is the sensitivity of our results to data migration. Figure 8 and Figure 9 show the effects of the bank data migration on execution cycle and energy results, respectively, for the LAB prediction scheme. We see that, for the three benchmarks with low L1 miss rates (mgrid, parser and gap) the migration scheme has little impact on the benchmark performance. The art benchmark has a 14% L1 miss rate and performance increases due to migration, as expected from the work by Kim et al. [1]. However, we also observe that this increase in performance is accompanied by a significant increase in average energy consumption per access. This increase is caused by the additional data

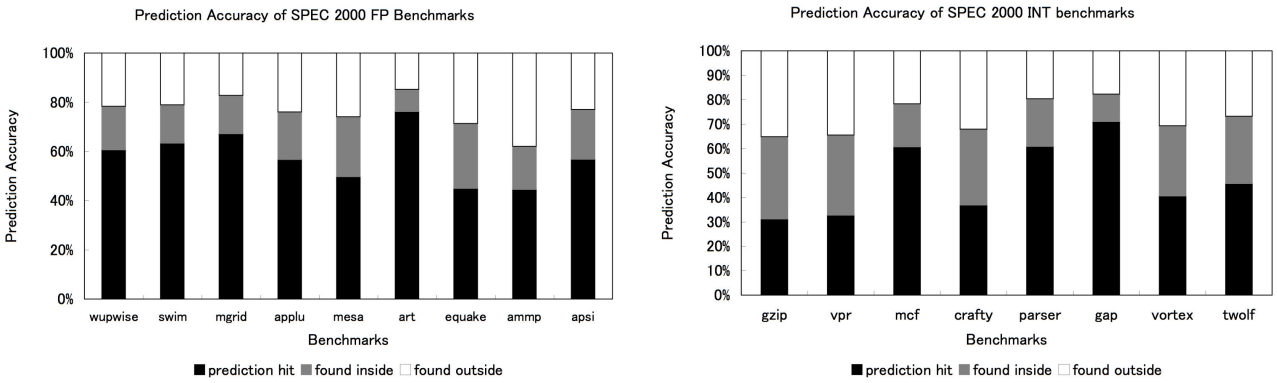


Fig. 3. Prediction accuracies for the SPEC CPU2000 benchmarks.

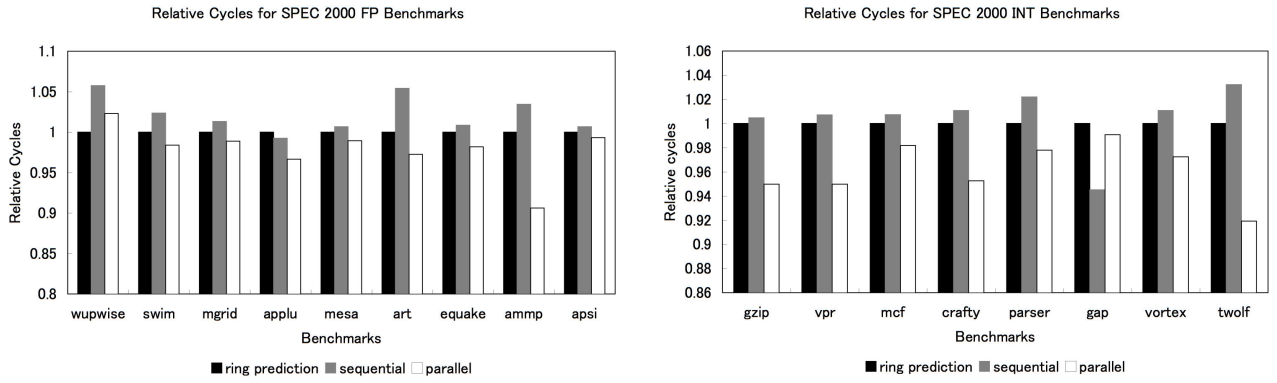


Fig. 4. Execution cycles for the SPEC CPU2000 benchmarks with our ring prediction based scheme, sequential scheme, and parallel scheme. For each benchmark, the last two bars are normalized with respect to the first bar.

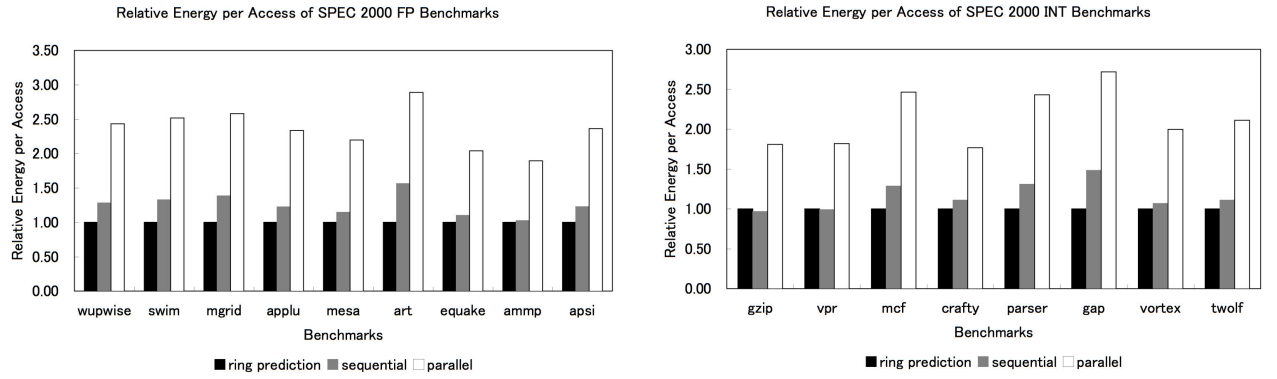


Fig. 5. Relative energy per access of SPEC CPU2000 benchmarks with our ring prediction based scheme, sequential scheme, and parallel scheme. For each benchmark, the last two bars are normalized with respect to the first bar.

movements inside of the cache structure.

V. Related Work

In this section, we discuss related work on NUCA based architectures. Kim et al. [1] introduced the concept of Non-Uniform Cache Architecture (NUCA), based on the observation that the increasing wire delay dominates the L2 cache access latency. Thus the access latency to a large L2 cache becomes dependent on the distance between the processor and the cache bank being accessed. Based on this observation, Kim et al. designed several NUCA architectures, by partitioning L2 cache into multiple banks and using a switched network to connect these banks. They proposed associated cache management policies such as cache line placement, search, and migration. Incremental and multicast searches are two methods commonly used to locate a cache line. Smart search, which is based on partial tag comparison, reduces the L2 miss determination time and the number of L2 bank lookups. Our work, based on ring prediction, proposes alternate cache line search policies between the two

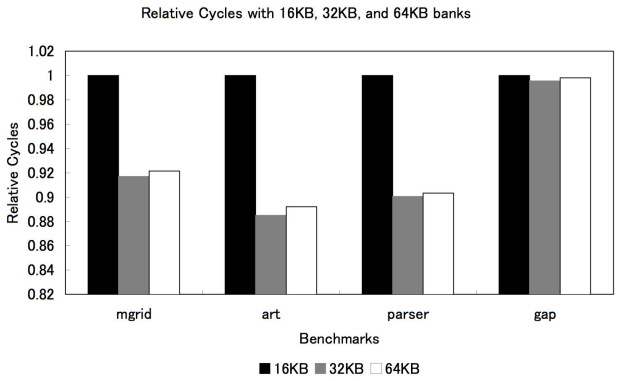


Fig. 6. Impact of bank size on execution cycles.

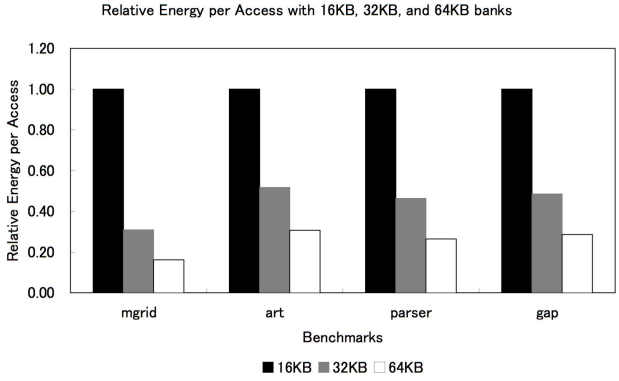


Fig. 7. Impact of bank size on energy consumption.

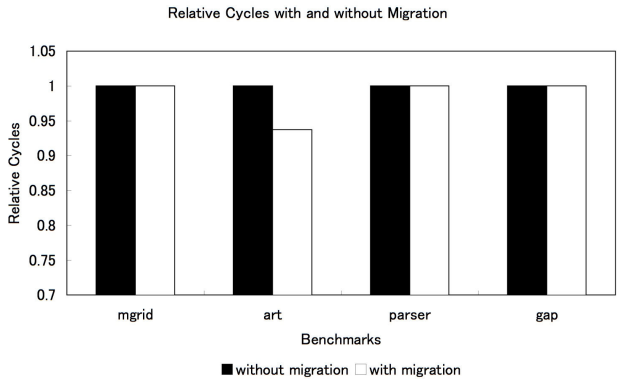


Fig. 8. Execution cycles with our LAB ring prediction scheme with and without data migration.

extreme search policies: incremental and multicast. The goal is to provide a better performance-power trade-off for cache line searches in NUCA. Note that our schemes can work independently of or together with smart search, which mainly optimizes L2 miss behavior. Chishti et al. [12] proposes another NUCA architecture, which features a more flexible cache placement and sequential tag-data accesses. However, that design requires that additional pointers from the tag array to the data array be maintained and updated by the cache.

Since emerging multi-core architectures often include a

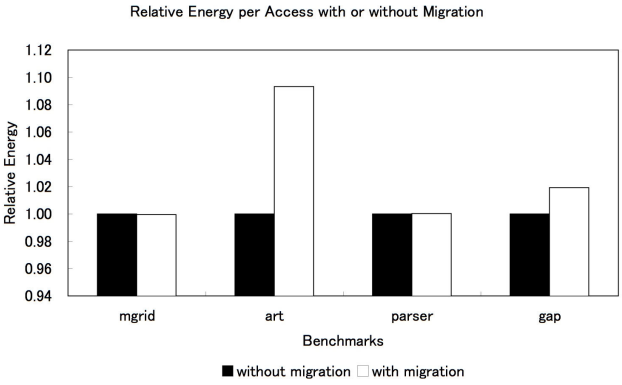


Fig. 9. Relative energy per access with the LAB ring prediction scheme with and without data migration.

large on-chip L2 cache, many research efforts have devoted into NUCAs under Chip Multiprocessor (CMP) scenarios. [13], [14], [2], [15], [16], [17], [18], [19] are the representative works belonging to this category. Most of these works targeted the trade-offs between short L2 access latency and large effective L2 capacity based on data replication, migration or cache partitioning techniques. Little attention is paid to cache line search policies. Normally they employ incremental/multicast searches or some cache coherence policy, to locate a cache line in an L2 NUCA. All of them targeted L2 cache performance, without considering L2 power optimization. In this paper, we focus on uniprocessor architecture, in order to better understand impacts of cache line search strategy on both performance and power. Our work is important in light of processor designs, where processor cores are surrounded by many banks (Guz et al. [20] and Liu et al. [21]).

Both cache and Network-on-Chip (NoC) consume significant portions of total chip power. There have been enormous efforts aiming to optimize cache or NoC energy consumption. Notable works around cache energy optimization are cache decay [22] and drowsy cache [23], which turn off or scale down the voltages of inactive cache lines based on cache access patterns. Regarding NoC energy optimization, several software- and hardware-based techniques [24], [25], [26] are proposed. However, to the best of our knowledge, no previous work optimized the power consumption of an NoC connected NUCA architecture. Our work aims to reduce the power consumption of both caches and NoCs in an NoC-based NUCA simultaneously by controlling the number of cache lookups and the number of multicast requests through search policies.

Development of NUCA cache architecture has also spurred development of unique hybrid private/shared caches. Guz et al. [20] proposed their multi-core architecture Nahalal, where the shared elements of the cache are located at the center of the die, while private elements of the cache are located on the outside and periphery of the system die. Liu et al. [21] consider utilizing a single small block of the NUCA cache to serve as a shared data block for all

processors. The ever decreasing feature sizes, and increasing wire latencies are also forcing designers to consider on and off chip interconnect properties in designing the coherency protocols [3], [27], as well as moving into three dimensions in order to reduce the average hop count [4].

VI. Conclusions

In this paper, we propose and experimentally evaluate the Last Access Based (LAB) ring prediction data access scheme for NUCA based systems. We compare our scheme against two base access schemes: the parallel access scheme and sequential access scheme. Our LAB scheme uses the last accessed ring to predict the L2 ring that holds the data needed on current request. Our LAB ring prediction mechanism, results in significant energy savings over the fully parallel and fully sequential schemes. Our LAB scheme reduces energy consumption by 15.9% relative to the sequential lookup scheme, and 53.8% relative to the parallel lookup scheme. Our ongoing work includes exploration of prediction aware data replication schemes within NUCA, and novel ring prediction schemes. We are also evaluating effectiveness of our ring predictor on other types of applications.

References

[1] C. Kim, D. Burger, and S. Keckler, "An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches," in *Proc. the International Conference on Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[2] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing," in *Proc. the International Conference on Supercomputing*, 2005.

[3] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. B. Carter, "Interconnect-aware coherence protocols for chip multiprocessors," *SIGARCH Comput. Archit. News*, vol. 34, no. 2, pp. 339–351, 2006.

[4] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, and M. Kandemir, "Design and management of 3d chip multiprocessors using network-in-memory," in *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, 2006, pp. 130–141.

[5] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das, "A novel dimensionally-decomposed router for on-chip communication in 3d architectures," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 138–149, 2007.

[6] J. A. Brown, R. Kumar, and D. Tullsen, "Proximity-aware directory-based coherence for multi-core processor architectures," in *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, 2007, pp. 126–134.

[7] N. Muralimanohar and R. Balasubramonian, "Interconnect design considerations for large nuca caches," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 369–380, 2007.

[8] S. P. E. Corporation, "Spec cpu2000," http://www.spec.org/cpu/.

[9] virtutech, "Virtutech simics," http://www.virtutech.com.

[10] D. Tarjan, S. Thoziyoor, and N. P. Jouppi, "Cacti 4.0," HP Laboratories Palo Alto, Tech. Rep., 2006.

[11] X. Chen and L.-S. Peh, "Leakage power modeling and optimization in interconnection networks," in *Proc. International Symposium on Low Power Electronics and Design 2003 (ISLPED2003)*, 2003.

[12] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Distance associativity for high-performance energy-efficient non-uniform cache architectures," in *Proc. the annual IEEE/ACM International Symposium on Microarchitecture*, 2003.

[13] B. M. Beckmann and D. A. Wood, "Managing wire delay in large chip-multiprocessor caches," in *Proc. the 37th annual IEEE/ACM International Symposium on Microarchitecture*, 2004.

[14] Z. Chishti, M. D. Powell, and T. N. Vijaykumar, "Optimizing replication, communication, and capacity allocation in cmps," in *Proc. the 32nd Annual International Symposium on Computer Architecture*, 2005.

[15] A. Zhang and K. Asanovic, "Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors." in *Proc. the International Symposium on Computer Architecture*, 2005.

[16] B. M. Beckmann, M. R. Marty, and D. A. Wood, "ASR: Adaptive selective replication for CMP caches," in *Proc. the International Symposium on Microarchitecture*, 2006.

[17] J. Chang and G. S. Sohi, "Cooperative caching for chip multiprocessors," in *Proc. the International Symposium on Computer Architecture*, 2006.

[18] T. Y. Yeh and G. Reinman, "Fast and fair: data-stream quality of service," in *Proc. the International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, 2005.

[19] S. Kim, D. Chandra, and Y. Solihin, "Fair cache sharing and partitioning in a chip multiprocessor architecture," in *Proc. the International Conference on Parallel Architectures and Compilation Techniques*, 2004.

[20] Z. Guz, I. Keidar, A. Kolodny, and U. Weiser, "Nahalal: Cache organization for chip multiprocessors," in *Computer Architecture Letters*, vol. 6, pp. 21 – 24.

[21] C. Liu, A. Sivasubramaniam, M. Kandemir, and M. Irwin, "Enhancing l2 organization for cmps with a center cell," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, p. 10.

[22] S. Kaxiras, Z. Hu, and M. Martonosi, "Cache decay: exploiting generational behavior to reduce cache leakage power," in *Proc. the annual international symposium on Computer architecture*, 2001, pp. 240–251.

[23] K. Flautner, N. S. Kim, S. Martin, D. Blaauw, and T. Mudge, "Drowsy caches: simple techniques for reducing leakage power," in *Proc.the annual international symposium on Computer architecture*, 2002, pp. 148–157.

[24] V. Soteriou, N. Eisley, and L.-S. Peh, "Software-directed power-aware interconnection networks," in *Proc. International Conference on Compilers, Architectures and Synthesis of Embedded Systems*, 2005.

[25] V. Soteriou and L.-S. Peh, "Design space exploration of power-aware on/off interconnection networks," in *Proc. the 22nd Int. Conf. on Computer Design*, Oct. 2004.

[26] G. Chen, F. Li, M. Kandemir, and M. J. Irwin, "Reducing noc energy consumption through compiler-directed channel voltage scaling," in *Proc. the 2006 ACM SIGPLAN conference on Programming Language Design and Implementation*. New York, NY, USA: ACM Press, 2006, pp. 193–203.

[27] Z. Radović and E. Hagersten, "Efficient synchronization for nonuniform communication architectures," in *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, 2002, pp. 1–13.