NUCA: A Non-Uniform Cache Access Architecture for Wire-Delay Dominated On-Chip Caches

Changkyu Kim Doug Burger Stephen W. Keckler

The University of Texas at Austin

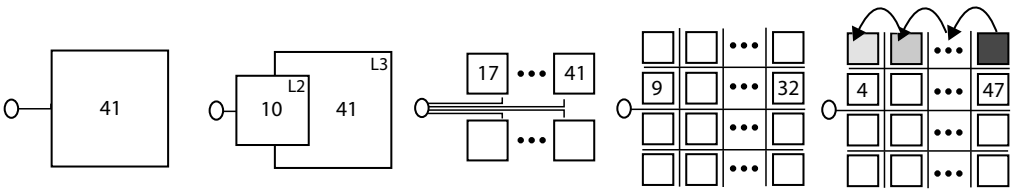
November 23, 2003

1 Introduction

This paper describes Non-Uniform Cache Access (NUCA) designs, which solve the on-chip wire delay problem for future large integrated caches. These designs embed a network into the cache itself, allowing data to migrate within the cache, clustering the working set in the cache region nearest to the processor.

Today's high performance processors incorporate large level-two (L2) caches on the processor die. The IBM Power5 will contain a 1.92MB L2 cache, the HP PA-8700 will contain 2.25MB of unified on-chip cache [5], and the Intel Itanium2 will contain 6MB of on-chip L3 cache. These sizes will continue to increase as the bandwidth demands on the package grow, and as smaller technologies permit more bits per mm^2 [6]. In future technologies, large on-chip caches with a single, discrete hit latency will be undesirable, due to increasing global wire delays across the chip [1, 10]. Data residing in the part of a large cache close to the processor could be accessed much faster than data that reside physically farther from the processor. The closest bank in a 16-megabyte, on-chip L2 cache built in a 50-nanometer process technology could be accessed in 4 cycles, while an access to the farthest bank might take 47 cycles. The bulk of the access time will involve routing to and from the banks, not the bank accesses themselves.

Figure 1 shows the types of organizations that we explore in this paper, listing the number of banks and the average access times, assuming 16MB caches modeled with a 50nm technology. The numbers superimposed



	(a) UCA	(b) ML-UCA	(c) S-NUCA-1	(d) S-NUCA-2	(e) D-NUCA
Number of banks:	1 bank	8/32 banks	32 banks	32 banks	256 banks
Avg. loaded access time:	*255 cycles*	*11/41 cycles*	*34 cycles*	*24 cycles*	*18 cycles*

Figure 1: Level-2 Cache Architectures

on the cache banks show the latency of a single contentionless request. The average loaded access times are derived from performance simulations that use the unloaded latency as the access time but which include port and channel contention.

We call a traditional cache a Uniform Cache Architecture (UCA), shown in Figure 1a. Although we support aggressive sub-banking, our models indicate that this cache would perform poorly due to internal wire delays and restricted numbers of ports. Figure 1b shows a traditional, but aggressively banked multi-level cache (L2 and L3), called ML-UCA. Inclusion is enforced, so a line in the smaller level implies two copies in the cache, consuming extra space. Figure 1c shows a cache that supports non-uniform access to the different banks without the inclusion overhead of ML-UCA. The mapping of data into banks is statically determined, based on the block index, and thus can reside in only one bank of the cache. Each bank uses a private, two-way, pipelined transmission channel to service requests. We call this statically mapped, non-uniform cache S-NUCA-1.

When the delay to route a signal across a cache is significant, increasing the number of banks can improve performance. However, private per-bank channels, used in S-NUCA-1, heavily restrict the number of banks that can be implemented, since the private channel wires add significant area overhead if the number of banks is large. To circumvent that limitation, we propose a static NUCA design with a two-dimensional switched network instead of private per-bank channels, permitting a larger number of smaller, faster banks. This organization, called S-NUCA-2, is shown in Figure 1d.

Even with an aggressive multi-banked design, performance may still be improved by exploiting the fact that accessing closer banks is faster than accessing farther banks. By permitting data to be mapped to one of many banks within the cache, and to migrate among them, a cache can be automatically managed in such a way that most requests are serviced by the fastest banks. Using the switched network, data can be gradually promoted to faster banks as they are frequently used. We call this dynamic non-uniform cache D-NUCA, which is depicted in Figure 1e.

2 Statically Mapped NUCA Caches

2.1 Experimental Methodology

To evaluate the effects of different cache organizations on system performance, we used Cacti 3.0 [14] to derive the access times for caches, and extended the sim-alpha simulator [3] which models the Alpha 21264 in detail to simulate different cache organizations. For the rest of this paper, we assume a constant L2 cache area and vary the technology generation to scale cache capacity within that area, using the ITRS Roadmap [13] predictions. The benchmarks used in our study include six SPEC2000 floating-point benchmarks, six SPEC2000 integer benchmarks, three scientific applications from the NAS suite, and Sphinx, a speech recognition application. For each benchmark we simulated the sequence of instructions which capture the core repetitive phase of the program, determined by plotting the L2 miss rates over one execution of each benchmark, and choosing the smallest subsequence that captured the recurrent behavior of the benchmark.

Technology	L2	Num. Banks			Unloaded Avg.			Loaded Avg.			IPC		
(nm)	Size	UCA	SNUCA1	SNUCA2	UCA	SNUCA1	SNUCA2	UCA	SNUCA1	SNUCA2	UCA	SNUCA1	SNUCA2
130	2MB	1	16	16	13	10	8	68	10	10	0.41	0.55	0.55
100	4MB	1	32	32	18	15	10	91	15	12	0.39	0.57	0.58
70	8MB	1	32	32	26	19	18	144	19	20	0.34	0.63	0.62
50	16MB	1	32	32	41	29	21	255	30	24	0.26	0.62	0.65

Table 1: Statically mapped NUCA performance

2.2 UCA Caches

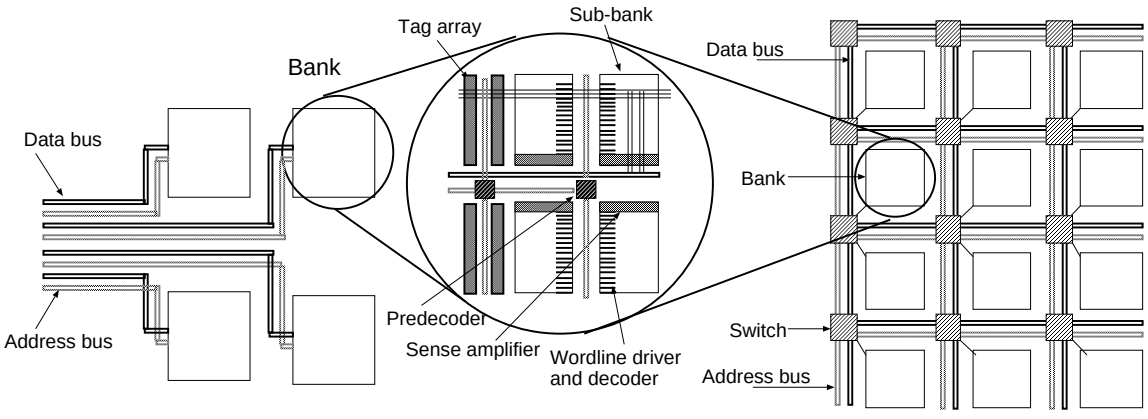
Table 1 shows the parameters and achieved instructions per cycle (IPC) of the three statically mapped cache organizations: `UCA`, `S-NUCA-1`, and `S-NUCA-2`. In Table 1, the unloaded latency is the average access time (in cycles) assuming uniform bank access distribution and no contention. The loaded latency is obtained by averaging the actual L2 cache access time–including contention–across all the benchmarks. Contention can include both *bank contention*, when a request must stall because the needed bank is busy, and *channel contention*, when the bank is free but the routing path to the bank is busy. The table shows the best cache configuration for each capacity along with the harmonic mean IPC across all the benchmarks.

In the UCA cache, the unloaded access latencies are sufficiently high that contention is a serious problem. Multiported cells are a poor solution for overlapping accesses in large caches, as increases in area will expand loaded access times significantly. Table 1 shows that, despite the aggressive cache pipelining, the loaded latency grows significantly as the cache size increases. The best overall cache size is 2MB; for larger caches, the continued reduction in L2 misses does not overcome the increases in latency. While the UCA organization is inappropriate for large, wire-dominated caches, it serves as a baseline for measuring the performance improvement of more sophisticated cache organizations.

2.3 Static NUCA Caches (S-NUCA-1, S-NUCA-2)

Multiple banks can mitigate the performance losses arising from worst-case uniform access latency, if each bank can be accessed at different speeds, proportional to the distance of the bank from the cache controller. Data are statically mapped into banks, with the low-order bits of the index determining the bank. Each bank we simulate is four-way set associative. These static, non-uniform cache architectures (`S-NUCA`) have two advantages over the `UCA` organization. First, accesses to banks closer to the cache controller incur lower latency. Second, accesses to different banks may proceed in parallel, reducing contention.

As shown in Figure 2a, each addressable bank in the `S-NUCA-1` organization has two private, per-bank 128-bit channels, one going in each direction. Such a design is used in the IBM Power4 level-2 cache. We used Cacti 3.0 along with the more aggressive repeater and scaled wire model of Agarwal *et al.* for the address and data busses to and from the banks [1]. Since banks have private channels, each bank can be accessed independently. While smaller banks would provide more concurrency and a greater fidelity of non-uniform access, the numerous per-bank channels add area overhead to the array that constrains the number of banks. Table 1 shows that, unlike `UCA`, the average IPC increases as the cache sizes increases until 8 MB. At 16MB, the cross-cache routing delays required by the cache causes the hit latencies to overwhelm the performance benefit of the reduced misses. After 4MB, the optimal number of banks does not increase further, due to the area overhead of the per-bank channels, making each bank larger and slower as the cache size increases. This constraint prevents the `S-NUCA-1` organization from exploiting the potential access



(a) S-NUCA-1 cache design (b) S-NUCA-2 cache design

Figure 2: S-NUCA Caches

fidelity of small, fast banks.

Figure 2b shows an organization that removes most of the wires resulting from per-bank channels. This organization, called `S-NUCA-2`, embeds a lightweight, wormhole-routed 2-D mesh with point-to-point links in the cache, placing simple switches at each bank. Each link has two separate 128-bit channels for bidirectional routing. We used the delays from a detailed circuit simulator in our cycle-accurate model of the switched network for our performance evaluation.

As shown in Table 1, for 4MB and larger caches, the unloaded latencies are smaller than those for `S-NUCA-1`. The switched network speeds up cache accesses because it consumes less area than the private, per-bank channels, resulting in a smaller array and faster access to all banks (at 50nm, a factor of 4 lower). Furthermore, the loaded latencies of the `S-NUCA-2` are always lower than the `S-NUCA-1`, providing a 10% IPC improvement for a 16MB cache.

3 Dynamic NUCA Implementations

In this section, we show how to exploit future cache access non-uniformity by placing frequently accessed data in closer banks and less important–yet still cached–data in farther banks. We evaluate a number of hardware policies that migrate data among the banks. For these policies, we answer three important questions about the management of data in the cache: (1) *mapping*: how the data are mapped to the banks, and in which banks a datum can reside, (2) *search*: how the set of possible locations are searched to find a line, (3) *movement*: under what conditions the data should be migrated from one bank to another.

3.1 Logical to Physical Cache Mapping

A large number of banks provides substantial flexibility for mapping lines to banks. At one extreme is the S-NUCA strategy, mapping a line of data to a single statically determined bank. At the other extreme,

a line could be mapped into any cache bank. While the latter approach maximizes placement flexibility, the overhead of locating the line may be too large. We explore an intermediate solution called *spread sets* in which the multibanked cache is treated as a set-associative structure, each set is spread across multiple banks, and each bank holds one (or more) "ways" of the set. The collection of banks used to implement this associativity is called a *bank set* and the number of banks in the set corresponds to the associativity. The primary distinction between this organization and a traditional set-associative cache is that the different associative ways have different access latencies.

In this paper, we evaluate one methods of allocating banks to bank sets and ways, called *simple mapping*. With the simple mapping, each column of banks in the cache becomes a bank set, and all banks within that column comprise the set-associative ways. The two drawbacks of this scheme are that the number of rows may not correspond to the number of desired ways in each bank set, and that latencies to access all bank sets are not the same. Detailed descriptions of two other mapping policies are presented in other work [9].

3.2 Locating Cached Lines

A distributed cache array, in which the tags are distributed with the banks, creates two new challenges. First, many banks may need to be searched to find a line on a cache hit. Second, if the line is not in the cache, the slowest bank determines the time necessary to resolve that the request is a miss. Thus, the miss resolution time grows as the number of banks in the bank set increases.

Searching for a line among a bank set can be done with two distinct policies. The first is *incremental search*, in which the banks are searched in order starting from the closest bank until the requested line is found or a miss occurs in the last bank. The second policy is called *multicast search*, in which the requested address is multicast to some or all of the banks in the requested bank set. However, both policies cannot deal with the above two challenges at the same time.

To further reduce both the number of bank lookups and the miss resolution time, we applied the idea of the *partial tag comparison* proposed by Kessler et al. [8]. The D-NUCA policy using partial tag comparisons, which we call *smart search*, stores the partial tag bits into a smart search array located in the cache controller. We evaluated two smart search policies: *ss-performance* and *ss-energy*. In the *ss-performance* policy, the cache array and the smart search array are searched in parallel, and if no matches occur in the smart search array, miss processing commences early. In the *ss-energy* policy, the partial tag comparison is used to reduce the number of banks that are searched upon a miss. To hide the delay to access smart search array (4 to 6 cycles), the cache controller searches the first cache bank in parallel with the smart search array.

3.3 Dynamic Movement of Lines

Since the goal of the dynamic NUCA approach is to maximize the number of hits in the closest banks, a desirable policy would be to make the closest bank hold the MRU line, second closest hold second most-recently used, etc. The problem with that approach is that most accesses would result in heavy movement of lines among banks. We use *generational promotion* [4] to balance the increased contention and power consumption of copying with the benefits expected from bank set ordering. When a hit occurs to a cache line, it is swapped with the line in the bank that is the next closest to the cache controller. Heavily used lines will thus migrate toward close banks, whereas infrequently used lines will be demoted into farther banks.

A D-NUCA policy must determine the placement of an incoming block resulting from a cache miss. A

Configuration	Technology (nm)	L2 Size	Num. Banks	Loaded Avg.	IPC	Miss Rate	Bank Accesses	Tag Bits	Search Array
Base D-NUCA	130	2MB	16	8.4	0.57	0.23	73M	-	—
	100	4MB	32	10.0	0.63	0.19	72M	-	—
	70	8MB	128	15.2	0.67	0.15	138M	-	—
	50	16MB	256	18.3	0.71	0.11	266M	-	—
SS-energy + shared bank	50	16MB	256	19.2	0.75	0.11	47M	6	216KB
Upper bound	50	16MB	256	3.0	0.83	0.114	—	-	—
Upper bound + SS-performance	50	16MB	256	3.0	0.89	0.114	—	7	224KB

Table 2: D-NUCA performance

replacement may be loaded close to the processor, displacing an important existing block. The replacement may be loaded in a distant bank, in which case an important new block would require several accesses before it is eventually migrated to the fastest banks. Another policy decision involves what to do with a victim upon a replacement; the two polices we evaluated were one in which the victim is evicted from the cache (a *zero-copy* policy), and one in which the victim is moved to a lower-priority bank, replacing a less important line farther from the controller (*one-copy* policy).

4 Performance Evaluation

4.1 Policy Exploration

The first four rows of Table 2 shows the performance of the baseline `D-NUCA` configuration, which uses the simple mapping, multicast search, tail insertion, and single-bank promotion upon each hit. As the capacities increase with the smaller technologies, from 2MB to 16MB, the IPC gains over the best of the `S-NUCA` grows from 4% to 9%.

The next three rows of Table 2 shows the efficacy of the smart search policy at improving IPC *and* reducing bank accesses. With the SS-energy policy, a reduction of 85% of the bank lookups can be achieved, with a 6% IPC gain over the base `D-NUCA` configuration. Coupling the SS-energy policy with the shared mapping, tail insertion, and single-bank promotion upon each hit results in the best policy we examined. We call this policy the "best" `D-NUCA` policy, `DN-best`, since it balances high performance with a relatively small number of bank accesses. An exploration of the policy space and a more rigorous analysis is presented in [9]. The last two rows of Table 2 shows two upper bounds on IPC. The first upper bound row shows the mean IPC that would result if all accesses hit in the closest bank with no contention. The second row shows the same metric, but with early initiation of misses provided by the smart search array. The upper bound is 19% better than the `DN-best` policy.

4.2 Comparison to ML-UCA

Multi-level hierarchies permit a subset of frequently used data to migrate to a smaller, closer structure similar to a `D-NUCA` cache, but at a coarser grain than individual banks. We compared the `NUCA` schemes with a two-level hierarchy (L2 and L3), called `ML-UCA`. We modeled the L2/L3 hierarchy by assuming that both levels were aggressively pipelined and banked `UCA` structures. We also assumed that the L3 had the same

Technology (nm)	L2/L3 Size	Num. Banks	Unloaded Latency	Loaded Latency	ML-UCA IPC	DN-best IPC
130	512KB/2MB	4/16	6/13	7.1/13.2	0.55	0.58
100	512KB/4MB	4/32	7/21	8.0/21.1	0.57	0.63
70	1MB/8MB	8/32	9/26	9.9/26.1	0.64	0.70
50	1MB/16MB	8/32	10/41	10.9/41.3	0.64	0.75

Table 3: Performance of an L2/L3 Hierarchy

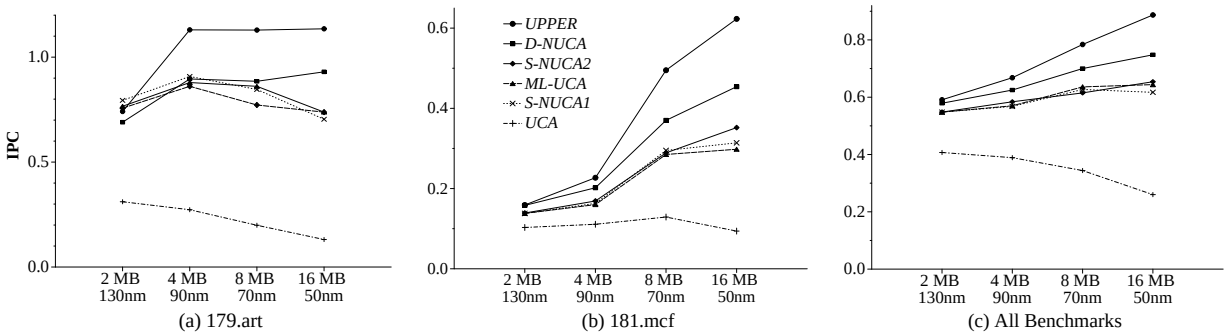


Figure 3: Performance summary of major cache organizations

size as the comparable NUCA cache, and chose the L2 size and L3 organization that maximized overall IPC. The ML-UCA organization thus consumes more area than the single-level L2 caches, and has a greater total capacity of bits. In addition, we assumed no additional routing penalty to get from the L2 to the L3 upon an L2 miss, essentially assuming that the L2 and the L3 reside in the same space, making the multi-level model optimistic.

Table 3 compares the IPC of the ideal two-level ML-UCA with a D-NUCA cache. In addition to the optimistic ML-UCA assumptions listed above, we assumed that the two levels were searched in parallel upon every access. The IPC of the two schemes is roughly comparable at 2MB, but diverges as the caches grow larger. At 16MB, overall IPC is 17% higher with DN-best than with the ML-UCA, since many of the applications have working sets greater than 2MB, incurring unnecessary misses, and some have working sets smaller than 2MB, rendering the ML-UCA L2 too slow.

The two designs compared in this subsection are not the only points in the design space. For example, one could view a simply-mapped D-NUCA as an n-level cache (where n is the bank associativity) that does not force inclusion, and in which a line is migrated to the next highest level upon a hit, rather than the highest. A D-NUCA cache could be designed to permit limited inclusion, supporting multiple copies within a spread set. Alternatively, a ML-UCA in which the two (or more) levels were each organized as S-NUCA-2 designs, and in which inclusion was not enforced, would start to resemble a D-NUCA organization in which lines could only be mapped to two places. However, our experiments with many D-NUCA policies indicate that the ability to effectively adjust the size of the active working set, clustered near the processor, provides better performance and performance stability than competing alternatives.

4.3 Cache Design Comparison

Figure 3 shows how the various schemes perform across technology generations and thus cache sizes. The IPC of *art*, with its small working set size, is shown in Figure 3a. Figure 3b shows the same information for a benchmark (*mcf*) that has a larger-than-average working set size. Figure 3c shows the harmonic mean IPC across all benchmarks.

First, the IPC improvements of D-NUCA over the other organizations grows as the cache grows larger. The adaptive nature of the D-NUCA architecture permits consistently increased IPC with increased capacity, even in the face of longer wire and on-chip communication delays. Second, the D-NUCA organization is *stable*, in that it makes the largest cache size the best performer for twelve among sixteen applications we examined. Figure 3a shows this disparity most clearly in that D-NUCA is the only organization for which *art* showed improved IPC for caches larger than 4MB.

5 Summary and Conclusions

This work is the first to propose novel designs for large, wire-dominated on-chip caches, but significant prior work has evaluated large cache designs. Kessler examined designs for multi-megabyte caches built with discrete components [7]. Hallnor and Reinhardt [4] studied a fully associative software-managed design for large on-chip L2 caches, but not did not consider non-uniform access times. Powell et al. evaluate the balance between incremental searches of the sets to balance power and performance [12], as NUCA caches do with multicast versus incremental policies, and as Kessler et al. did to optimize for speed [8]. Other researchers examined using multiple banks for high bandwidth, just as this work did to reduce contention. Sohi and Franklin [15] proposed interleaving banks to create ports, and also examined the port demand of L2 cache on less powerful processors than today's. Many researchers have also examined adaptive cache policies, a concept which is inherent in the D-NUCA organization. A good example is Dahlgren et al., who studied creative ways to avoid conflicts in direct-mapped on-chip caches by virtually binding regions of the address space to portions of the cache [2].

Although this work is the first to propose cache designs specifically targeted at wire delay scalability, non-uniform accesses have already started to appear in high performance cache designs [11]. This paper proposes several new designs that treat the cache as a network of banks and facilitates non-uniform accesses to different physical regions. We have shown that these non-uniform cache access (NUCA) architectures achieve the following three goals:

- *Low latency access*: the best 16MB D-NUCA configuration, simulated with projected 50nm technology parameters, demonstrated an average access time of 17 cycles at an 8 FO4 clock, which is a lower absolute latency than conventional L2 caches.
- *Technology scalability*: Increasing wire delays will increase access times for traditional, uniform access caches. The D-NUCA design scales much better with technology than conventional caches, since most accesses are serviced by close banks, which can be kept numerous and small due to the switched network.
- *Performance stability*: The ability of a D-NUCA cache to migrate data eliminates the trade-off between larger, slower caches for applications with large working sets and smaller, faster caches for

applications that are less memory intensive.

- *Flattening the memory hierarchy*: The D-NUCA design outperforms multi-level caches built in an equivalent area, since the multi-level cache has fixed partitions that are slower than an individual bank. This D-NUCA result augurs a reversal of the trend of deepening memory hierarchies. We foresee future memory hierarchies having two or at most three levels: a fast L1 tightly coupled to the processor, a large on-chip NUCA L2, and perhaps an off-chip L3 that uses a memory device technology other than SRAM. Future work will examine a further flattening of the entire cache hierarchy into a single NUCA structure.

Maintaining coherence among multiple NUCA caches presents new challenges. A variant of the partial tag compare scheme of Kessler et al. [8] may make distributed snooping economical. Emerging chip multiprocessors (CMP) architectures will likely benefit from the flexibility and scalability of NUCA memory systems. A natural organization places multiple processors and an array of cache banks on a single die. As the workload changes, NUCA cache banks can be dynamically partitioned and reallocated to different processors. Since the banks are individually addressable, the memory system may be reconfigured to support different programming models–such as streaming or vector workloads–by sending configuration commands to individual banks.

Finally, while the emergence of non-uniform cache latencies creates difficulties for some traditional optimization techniques, such as load-use speculation or compiler scheduling, we view the emergence of non-uniform accesses as inevitable. Those optimization techniques must be augmented to handle the non-uniformity where possible, or simply discarded where not.

Acknowledgments

This research is supported by the Defense Advanced Research Projects Agency under contract F33615-01-C-1892, NSF CAREER grants CCR-9985109 and CCR-9984336, two IBM University Partnership awards, and a grant from the Intel Research Council.

References

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate vs. IPC: The end of the road for conventional microprocessors. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*, pages 248–259, June 2000.

[2] F. Dahlgren and P. Stenström. On reconfigurable on-chip data caches. In *Proceedings of the 24th International Symposium on Microarchitecture*, pages 189–198, Nov. 1991.

[3] R. Desikan, D. Burger, S. W. Keckler, and T. Austin. Sim-alpha: A validated execution-driven alpha 21264 simulator. Technical Report TR-01-23, Department of Computer Sciences, University of Texas at Austin, 2001.

[4] E. Hallnor and S. Reinhardt. A fully associative software-managed cache design. In *Proceedings of the 27th International Symposium on Computer Architecture*, pages 107–116, June 2000.

[5] J. Hill and J. Lachman. A 900MHz 2.25 MB cache with on-chip CPU now in Cu SOI. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 171–177, February 2001.

[6] J. Huh, D. Burger, and S. W. Keckler. Exploring the design space of future CMPs. In *Proceedings of the 10th International Conference on Parallel Architectures and Compilation Techniques*, pages 199–210, September 2001.

[7] R. Kessler. *Analysis of Multi-Megabyte Secondary CPU Cache Memories*. PhD thesis, University of Wisconsin-Madison, December 1989.

[8] R. Kessler, R. Jooss, A. Lebeck, and M. Hill. Inexpensive implementations of set-associativity. In *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pages 131–139, May 1989.

[9] C. Kim, D. Burger, and S. W. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches. In *Proceedings ASPLOS-10*, pages 211–222, October 2002.

[10] D. Matzke. Will physical scalability sabotage performance gains? *IEEE Computer*, 30(9):37–39, September 1997.

[11] H. Pilo, A. Allen, J. Covino, P. Hansen, S. Lamphier, C. Murphy, T. Traver, and P. Yee. An 833MHz 1.5w 18Mb CMOS SRAM with 1.67Gb/s/pin. In *Proceedings of the 2000 IEEE International Solid-State Circuits Conference*, pages 266–267, February 2000.

[12] M. Powell, A. Agarwal, T. Vijaykumar, B. Falsafi, and K. Roy. Reducing set-associative cache energy via way-prediction and selective direct-mapping. In *Proceedings of the 34th International Symposium on Microarchitecture*, pages 54–65, December 2001.

[13] The international technology roadmap for semiconductors. Semiconductor Industry Association, 2001.

[14] P. Shivakumar and N. Jouppi. Cacti 3.0: An integrated cache timing, power and area model. Technical report, Compaq Computer Corporation, August 2001.

[15] G. Sohi and M. Franklin. High-performance data memory systems for superscalar processors. In *Proceedings of the Fourth Symposium on Architectural Support for Programming Languages and Operating Systems*, pages 53–62, Apr. 1991.

Author Info

- Changkyu Kim is a PhD student in Computer Sciences at UT-Austin. Kim has an MS in computer engineering from Seoul National University. He is a student member of the IEEE and ACM.
- Doug Burger is an Assistant Professor of Computer Sciences at UT-Austin. Dr. Burger has a PhD in computer science from the University of Wisconsin, is an Alfred P. Sloan Foundation fellow and a member of IEEE and ACM.
- Stephen W. Keckler is an Assistant Professor of Computer Sciences at UT-Austin. Dr. Keckler has a PhD in computer science from MIT, is an Alfred P. Sloan Foundation fellow and a member of IEEE and ACM.

Direct questions and comments about this article to Doug Burger, Department of Computer Sciences, The University of Texas at Austin, 1 University Station C0500, Austin, TX, 78712; dburger@cs.utexas.edu.