

---

# NONUNIFORM CACHE ARCHITECTURES FOR WIRE-DELAY DOMINATED ON-CHIP CACHES

---

NONUNIFORM CACHE ACCESS DESIGNS SOLVE THE ON-CHIP WIRE DELAY PROBLEM FOR FUTURE LARGE INTEGRATED CACHES. BY EMBEDDING A NETWORK IN THE CACHE, NUCA DESIGNS LET DATA MIGRATE WITHIN THE CACHE, CLUSTERING THE WORKING SET NEAREST THE PROCESSOR.

**Changkyu Kim**  
**Doug Burger**  
**Stephen W. Keckler**  
The University of Texas  
at Austin

..... The next generation of today's high-performance processors incorporate large level-two caches on the processor die. For example, the IBM Power5 will contain a 1.92-Mbyte L2 cache, the Hewlett-Packard PA8700 will contain 2.25 Mbytes of unified on-chip cache,<sup>1</sup> and the Intel Itanium2 will contain 6 Mbytes of on-chip L3 cache. Cache sizes will continue to increase as bandwidth demands on the package grow, and as smaller technologies permit more bits per square millimeter.<sup>2</sup> However, increasing global wire delays across the chip will make large on-chip caches with a single, discrete hit latency undesirable in future technologies.<sup>3,4</sup> Data residing near the processor in a large cache is much more quickly accessible than data residing far from the processor. Accessing the closest bank in a 16-Mbyte, on-chip L2 cache built in a 50-nm process technology, for example, could take four cycles, whereas accessing the farthest bank might take 47 cycles. The bulk of the access time involves routing to and from the banks rather than the bank accesses themselves.

Nonuniform cache access (NUCA) designs address this wire-delay problem. In this approach, a switched network allows data to migrate to different cache regions according to access frequency—that is, frequently accessed data migrates to areas closer to the processor. We propose several designs that treat the cache as a network of banks and facilitate nonuniform accesses to different physical regions. NUCA architectures offer low-latency access, increased scalability, and greater performance stability than conventional uniform access cache architectures.

## Cache design types

Figure 1 shows the cache organization types we explore in this article with the number of banks and average access times, assuming 16-Mbyte caches modeled with a 50-nm technology. The numbers on the cache banks represent the latency of a single contentionless request. We derived the average loaded access times from performance simulations that use the unloaded latency as the access

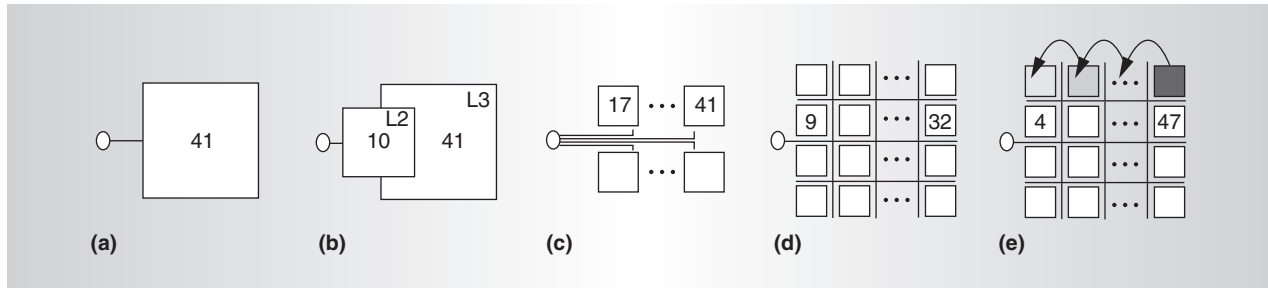


Figure 1. L2 cache architectures: uniform cache architecture (UCA) with one bank and a 255-cycle average loaded access time (a); multilevel UCA (ML-UCA) with 8 L2 and 32 L3 banks and 11- and 41-cycle average loaded access times for the two levels (b); static nonuniform cache architecture (S-NUCA-1) with 32 banks and a 34-cycle average loaded access time (c); static NUCA with a 2D switched network (S-NUCA-2) with 32 banks and a 24-cycle average loaded access time (d); and dynamic NUCA (D-NUCA) with 256 banks and an 18-cycle average loaded access time (e).

time but include port and channel contention.

Figure 1a shows a traditional cache, which we call a *uniform cache architecture* (UCA). Although we support aggressive subbanking, our models indicate that internal wire delays and restricted port numbers would cause this cache to perform poorly. Figure 1b shows a traditional but aggressively banked multilevel cache (ML-UCA). Inclusion is enforced, so a data line in the second level implies two copies in the cache, consuming extra space. Figure 1c shows a cache that supports nonuniform access to the different banks without the inclusion overhead of ML-UCA. This cache organization statically determines the mapping of data into banks based on the block index; thus data can reside in only one cache bank. Each bank uses a private, two-way, pipelined transmission channel to service requests. We call this statically mapped, nonuniform cache S-NUCA-1.

When the delay to route a signal across a cache is significant, partitioning the cache into more banks can improve performance. However, the private per-bank channels used in S-NUCA-1 add significant area overhead for large numbers of banks, thus restricting the number of banks into which the cache should be partitioned.

We propose the S-NUCA-2, shown in Figure 1d, a static NUCA design with a 2D switched network instead of private per-bank channels, permitting a larger number of smaller, faster banks.

Even with an aggressive multibanked design, we can still improve performance by exploiting the speed disparity between closer and

more distant banks. By permitting the cache controller to map data to any bank in the cache and migrate the data among the banks, it can automatically manage the cache such that the fastest banks service most requests. The switched network gradually promotes data to faster banks as accesses to that cache line continue. Figure 1e depicts this dynamic nonuniform cache, which we call D-NUCA.

### Statically mapped NUCA caches

To evaluate the effects of different cache organizations on system performance, we used the Cacti 3.0 model, a cache modeling tool,<sup>5</sup> to derive cache access times and extended the sim-alpha simulator,<sup>6</sup> which models the Alpha 21264 in detail to simulate different cache organizations. We assume a constant L2 cache area and vary the technology generation to scale cache capacity within it, using the *International Technology Roadmap for Semiconductors*<sup>7</sup> predictions. Our benchmarks include six SPEC2000 floating-point benchmarks, six SPEC2000 integer benchmarks, three scientific applications from the NAS (a scientific benchmark application) suite, and Sphinx, a speech recognition application. For each benchmark, we simulated the sequence of instructions that capture the program's core repetitive phase, which we determined by plotting the L2 miss rates over one execution of each benchmark and choosing the smallest sub-sequence capturing the benchmark's recurrent behavior.

### UCA caches

Table 1 shows the parameters and achieved

**Table 1. Statically mapped NUCA performance.**

Technology (nm)	L2 size (Mbytes)	Number of banks			Unloaded latency (average)			Loaded latency (average)			Instructions per cycle		
		UCA	S-	S-	UCA	S-	S-	UCA	S-	S-	UCA	S-	S-
			NUCA-1	NUCA-2		NUCA-1	NUCA-2		NUCA-1	NUCA-2		NUCA-1	NUCA-2
130	2	1	16	16	13	10	8	68	10	10	0.41	0.55	0.55
100	4	1	32	32	18	15	10	91	15	12	0.39	0.57	0.58
70	8	1	32	32	26	19	18	144	19	20	0.34	0.63	0.62
50	16	1	32	32	41	29	21	255	30	24	0.26	0.62	0.65

instructions per cycle (IPC) of the three statically mapped cache organizations: UCA, S-NUCA-1, and S-NUCA-2. In the table, the unloaded latency is the average access time (in cycles) assuming uniform bank access distribution and no contention. Loaded latency is the average actual L2 cache access time (including contention) across all benchmarks. Contention can include both *bank contention*—when a request stalls 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 of the IPC across all benchmarks.

In the UCA cache, the unloaded access latencies are high enough to make contention a serious problem. Multiported cells are a poor solution for overlapping accesses in large caches, as increases in area will significantly expand loaded access times. Table 1 shows that the loaded latency grows significantly as the cache size increases, despite the aggressive cache pipelining. The best overall cache size is 2 Mbytes; for larger caches, the continued reduction in L2 misses does not overcome latency increases. Although 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.

### Static NUCA caches

Multiple banks can mitigate performance losses arising from worst-case uniform access latency if a cache controller can access each bank at a speed proportional to its distance from the cache controller. The cache controller statically maps data to banks according to the index's low-order bits. Each bank we simulate is four-way set associative. These

S-NUCA organizations have two advantages over the UCA organization:

- Accesses to banks closer to the cache controller incur lower latency.
- Accesses to different banks can proceed in parallel, thereby reducing contention.

As Figure 2a shows, each addressable bank in the S-NUCA-1 organization has two private, per-bank 128-bit channels, one in each direction. The IBM Power4 L2 cache uses this design. We used Cacti 3.0 with Agarwal et al.'s more aggressive repeater and scaled-wire model<sup>3</sup> for the address and data busses to and from the banks. Because banks have private channels, the controller can access them independently. Although smaller banks provide more concurrency and greater nonuniform access fidelity, the numerous per-bank channels add area overhead to the array, constraining the number of banks. As Table 1 shows, the average IPC increases as cache size increases up to 8 Mbytes. At 16 Mbytes, the cross-cache routing delays cause the hit latencies to overwhelm the performance benefit of the reduced misses. After 4 Mbytes, the optimal number of banks stays the same because of the area overhead of the per-bank channels, making the banks larger and slower as cache size increases. This constraint prevents the S-NUCA-1 organization from exploiting the potential access fidelity of small, fast banks.

The S-NUCA-2 organization in Figure 2b removes most of the wires resulting from per-bank channels. It embeds a lightweight, wormhole-routed 2D 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. For our performance evaluation, we used the delays from a detailed circuit simulator in a cycle-accurate model of the switched network.

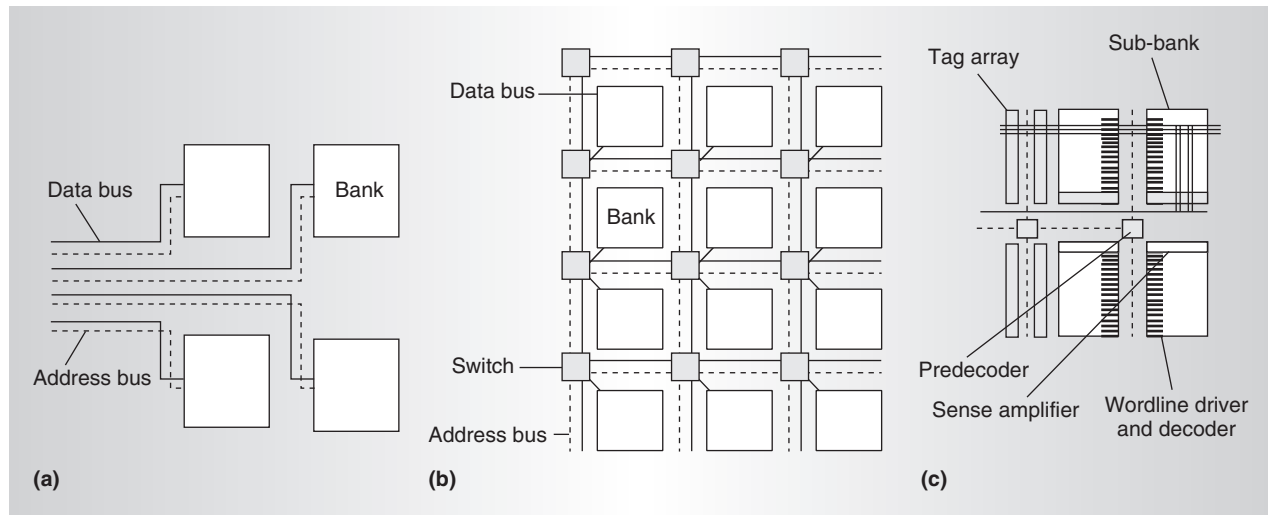


Figure 2. Static nonuniform cache architecture designs: S-NUCA-1 (a), S-NUCA-2 (b), and enlargement of a single bank (c).

As Table 1 shows, in 4-Mbyte and larger caches, the unloaded latencies for S-NUCA-2 are shorter than those for S-NUCA-1. Because it consumes less area than the private, per-bank channels, the switched network results in a smaller array (a factor of 4 smaller wire area overhead) and faster access to all banks. Furthermore, the loaded latencies of the S-NUCA-2 are always lower than those of the S-NUCA-1, providing a 10-percent IPC improvement for a 16-Mbyte cache.

### Dynamic NUCA implementations

We can exploit future cache access nonuniformity by placing frequently accessed data in closer banks and less important—but still cached—data in more distant banks. Several hardware policies migrate data between the banks. For these policies, we must address important questions about cache data management:

- *Mapping.* How are the data mapped to the banks? In which banks can a datum reside?
- *Search.* How does the cache controller search the set of possible locations to find a line?
- *Movement.* Under what conditions should the data migrate from one bank to another?

### Logical to physical cache mapping

Having many banks provides substantial

flexibility for mapping data lines to banks. The S-NUCA strategy is at one extreme, mapping a line to a single statically determined bank. Mapping a line to any cache bank is at the other extreme. Although this approach maximizes placement flexibility, locating the line might incur too large an overhead.

*Spread sets* are an intermediate solution. This approach treats the multibanked cache as a set-associative structure, with each set spread across multiple banks, and each bank holding one or more ways of the set. A *bank set* is the collection of banks that implement this associativity, which corresponds to the number of banks in the set. This organization differs from traditional set-associative caches primarily in that the different associative ways have different access latencies.

This article evaluates *simple mapping*, one method of allocating banks to bank sets and ways. With simple mapping, each column of banks in the cache becomes a bank set, with the banks in the column making up the set-associative ways. This scheme has two drawbacks:

- The number of rows might not correspond to the number of desired ways in each set.
- Latencies to access the sets differ.

Detailed descriptions of two other mapping policies appear elsewhere.<sup>8</sup>

### Locating cached lines

A distributed cache array, in which the tags are distributed with the banks, creates two new challenges. First, finding a line on a cache hit might require searching many banks. Second, if the line isn't in the cache, the slowest bank determines how long it will take to find out that the request is a miss. Thus, the miss resolution time grows as the number of banks in the bank set increases.

Two distinct policies permit searching for a line in a bank set. In an *incremental search*, the cache controller searches the banks in order, beginning with the closest bank and continuing until it finds the requested line or a miss occurs in the last bank. In multicast search the controller multicasts the requested address to some or all of the banks in the requested bank set. We can apply these two policies to solve one or the other of the two challenges posed by distributed cache arrays, but neither can alleviate both problems at the same time.

To further reduce both the number of bank lookups and the miss resolution time, we applied Kessler et al.'s idea of *partial tag comparison*.<sup>9</sup> The D-NUCA policy using partial tag comparisons, or *smart search*, stores partial tag bits in a smart-search array located in the cache controller. We evaluated two smart-search policies: *SS-performance* and *SS-energy*. The *SS-performance* policy involves searching the cache and smart-search arrays in parallel. If no matches occur in the smart-search array, miss processing commences early. The *SS-energy* policy uses the partial tag comparison to reduce the number of banks searched upon a miss. To hide the delay in accessing smart-search array (4 to 6 cycles), the cache controller searches the first cache bank in parallel with its search of the smart-search array.

### Dynamic line movement

Because the dynamic NUCA approach aims to maximize the number of hits in the closest banks, a desirable policy would make the closest bank hold the most recently used line, the second closest hold the second MRU line, and so on. However, in this approach, most accesses would cause heavy movement of lines between banks. We use *generational promotion*<sup>10</sup> to balance the increased contention and power consumption of copying with the ben-

efits expected from bank set ordering. When a hit occurs to a cache line, the cache controller swaps the cache line with the line in the next closest bank to the controller. Heavily used lines thus migrate toward banks near the controller, whereas infrequently used lines are demoted to more distant banks.

A D-NUCA policy must determine the placement of an incoming block resulting from a cache miss. It can load a replacement near the processor, displacing an important existing block. Or a new block could go into a distant bank; then important new blocks would require several accesses before migrating to the fastest banks.

Another policy decision involves what to do with a replacement block's victim. One policy we evaluated—a *zero-copy* policy—evicts the victim from the cache. Another policy—a *one-copy* policy—moves the victim to a lower-priority bank, replacing a less important line farther from the controller.

### Performance evaluation

Because the policy space is very large, we simulated a representative set of the possible configurations. We first present results of our policy exploration, and compare D-NUCA with the multi-level cache.

#### Policy exploration

The first four rows of Table 2 show the performance of the baseline D-NUCA configuration, which uses simple mapping, multicast search, tail insertion, and single-bank promotion upon each hit. With smaller technologies, the cache capacities increase—from 2 to 16 Mbytes—and the IPC improvements over the best of the S-NUCA-2 grow from 4 to 9 percent.

The next three rows of Table 2 show how the smart-search policy improves IPC and reduces bank accesses. The *SS-energy* policy enables a reduction of 85 percent of the bank lookups and a 6-percent IPC gain over the base D-NUCA configuration. Of the policies we examined, coupling the *SS-energy* policy with shared mapping, tail insertion, and single-bank promotion upon each hit gave the best results. We call it the best D-NUCA policy, or *DN-best*, because it balances high performance with relatively few bank accesses. Other work explores the policy space and pro-

Table 2. D-NUCA performance.

Configuration	Technology (nm)	L2 size (Mbytes)	Number of banks	Loaded latency (average) (ns)	IPC	Miss rate	Bank accesses (million)	Tag bits	Search array (Kbytes)
Base D-NUCA	130	2	16	8.4	0.57	0.23	73	NA	NA
	100	4	32	10.0	0.63	0.19	72	NA	NA
	70	8	128	15.2	0.67	0.15	138	NA	NA
	50	16	256	18.3	0.71	0.11	266	NA	NA
SS-energy + shared bank	50	16	256	19.2	0.75	0.11	47	6	216
Upper bound	50	16	256	3.0	0.83	0.114	NA	NA	NA
Upper bound + SS-performance	50	16	256	3.0	0.89	0.114	NA	7	224

Table 3. Performance of an L2/L3 hierarchy.

Technology (nm)	Size, L2/L3 (Mbytes)	No. of banks, L2/L3	Unloaded latency, L2/L3 (cycles)	Loaded latency, L2/L3 (cycles)	ML-UCA IPC	DN-best IPC
130	0.5/2	4/16	6/13	7.1/13.2	0.55	0.58
100	0.5/4	4/32	7/21	8.0/21.1	0.57	0.63
70	1/8	8/32	9/26	9.9/26.1	0.64	0.70
50	1/16	8/32	10/41	10.9/41.3	0.64	0.75

vides a more rigorous analysis.<sup>8</sup>

The last two rows of Table 2 show two upper bounds on IPC. The first upper-bound row shows the mean IPC resulting if all accesses hit in the closest bank with no contention. The second row shows the same metric, but with the smart-search array initiating misses early. The upper bound is 19 percent better than the DN-best policy.

#### Comparison to multilevel UCA

Multilevel 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 granularity than that of individual banks. We compared the NUCA schemes with ML-UCA, modeling ML-UCA with three assumptions that make the multilevel model optimistic:

- Both levels are aggressively pipelined and banked UCA structures.
- L3 is the same size as the comparable NUCA cache. We chose the L2 size and the 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 bit capacity.

- Getting from L2 to L3 upon an L2 miss incurs no additional routing penalty—that is, L2 and L3 reside in the same space.

Table 3 compares the IPC of the ideal two-level ML-UCA with that of the DN-best cache. In addition to our optimistic ML-UCA assumptions, we assumed that the cache controller searches the two levels in parallel upon each access. The two schemes' IPCs are roughly comparable at 2 Mbytes, but diverge as the caches grow larger. At 16 Mbytes, overall IPC is 17 percent higher with DN-best than with ML-UCA. This is because many of the applications have working sets larger than 2 Mbytes, incurring unnecessary misses, and some have working sets smaller than 2 Mbytes, rendering the ML-UCA L2 too slow.

These two designs are not the only points in the design space. For example, you 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 migrates to

the next highest level upon a hit, rather than to the highest level. We could design a D-NUCA cache to permit limited inclusion, supporting multiple copies within a spread set. Alternatively, an ML-UCA in which the two (or more) levels were each organized as S-NUCA-2 designs, and which does not enforce inclusion would start to resemble a D-NUCA organization in which lines could only map 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.

### Cache design comparison

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

As the cache grows, D-NUCA's IPC improves. The D-NUCA architecture's adaptive nature lets IPC consistently increase as capacity grows, even in the face of longer wire and on-chip communication delays. In addition, the D-NUCA organization is stable in that it makes the largest cache size the best performer for 12 of the 16 applications we examined. Figure 3a shows this disparity most clearly: D-NUCA is the only organization for which *art* showed improved IPC for caches larger than 4 Mbytes.

Although this work is the first to propose cache designs specifically targeted at wire-delay scalability, nonuniform accesses are already appearing in high-performance cache designs.<sup>11</sup>

D-NUCA's outperformance of multilevel cache designs augurs a reversal of the trend toward deepening memory hierarchies. We foresee future memory hierarchies with 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 further flattening of the entire cache hierarchy into a single NUCA structure.

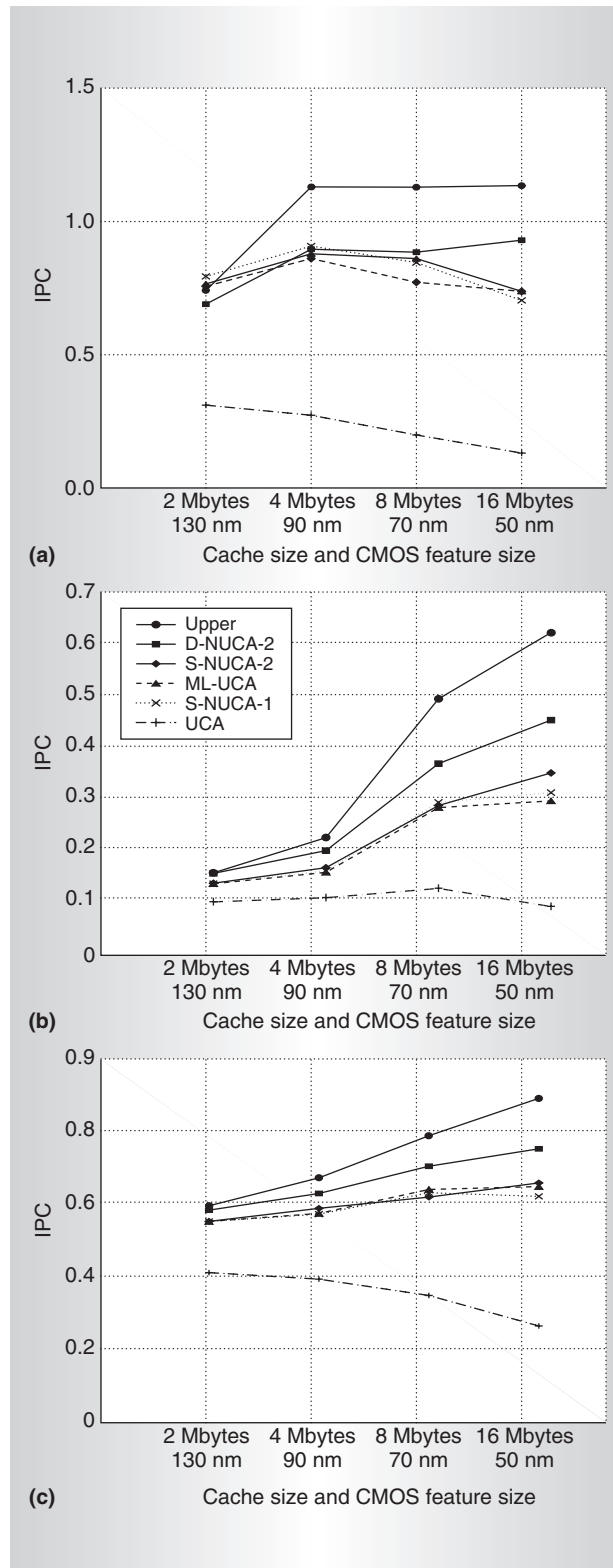


Figure 3. Performance summary of major cache organizations for 179.art (a) and 181.mcf (b) benchmarks, and the harmonic mean of all benchmarks (c).

## Related work on cache design

Although this article is the first to propose novel designs for large, wire-dominated on-chip caches, significant prior work has evaluated large cache designs.

Kessler examines designs for multimegabyte caches built with discrete components.<sup>1</sup> Hallnor and Reinhardt study a fully associative software-managed design for large on-chip L2 caches<sup>2</sup> but do not consider nonuniform access times.

Powell et al. evaluate the balance between incremental set searches to balance power and performance.<sup>3</sup> NUCA caches also tradeoff power and performance by balancing multicast versus incremental policies; Kessler et al. examine this same tradeoff between multicast and incremental search policies to optimize for speed.<sup>4</sup> Other researchers use multiple banks for high bandwidth; our goal was to reduce contention. Sohi and Franklin propose interleaving banks to create ports and examined the port demand of L2 cache on less powerful processors than today's.<sup>5</sup>

Many researchers have also examined adaptive cache policies, a concept inherent in the D-NUCA organization. A good example is Dahlgren et al., who avoid conflicts in direct-mapped on-chip caches by virtually binding regions of the address space to portions of the cache.<sup>6</sup>

## References

1. R. Kessler, *Analysis of Multi-Megabyte Secondary CPU Cache Memories*, PhD thesis, Univ. Wisconsin-Madison, Dec. 1989.
2. E. Hallnor and S. Reinhardt, "A Fully Associative Software-Managed Cache Design," *Proc. 27th Int'l Symp. Computer Architecture*, IEEE CS Press, 2000, pp. 107-116.
3. R. Kessler et al., "Inexpensive Implementations of Set-Associativity," *Proc. 16th Ann. Int'l Symp. Computer Architecture*, ACM Press, 1989, pp. 131-139.
4. M. Powell et al., "Reducing Set-Associative Cache Energy Via Way-Prediction and Selective Direct-Mapping," *Proc. 34th Int'l Symposium Microarchitecture*, IEEE CS Press, 2001, pp. 54-65.
5. G. Sohi and M. Franklin, "High-Performance Data Memory Systems for Superscalar Processors," *Proc. 4th Symp. Architectural Support for Programming Languages and Operating Systems*, ACM Press, 1991, pp. 53-62.
6. F. Dahlgren and P. Stenström, "On Reconfigurable On-Chip Data Caches," *Proc. 24th Int'l Symp. Microarchitecture*, ACM Press, 1991, pp. 189-198.

Maintaining coherence among multiple NUCA caches presents new challenges. A variant of Kessler et al.'s partial tag compare scheme<sup>9</sup> might make distributed snooping economical. Emerging chip multiprocessor (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, D-NUCA design can dynamically partition NUCA cache banks and reallocate them to different processors. Because the banks are individually addressable, we can reconfigure the memory system to support different programming models (such as streaming or vector workloads) by sending configuration commands to individual banks.

Finally, although the emergence of nonuniform cache latencies creates difficulties for some traditional optimization techniques, such as load-use speculation or compiler scheduling, nonuniform accesses are inevitable. Computer architects must therefore augment the optimization techniques to handle the nonuniformity where possible, or simply discard them.

MICRO

## Acknowledgments

The US 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 supported this research.

## References

1. J.M. Hill and J. Lachman, "A 900-MHz 2.25-Mbyte Cache with On-Chip CPU Now in SOI/Cu," *Proc. IEEE Int'l Solid-State Circuits Conf.*, IEEE Press, 2001, pp. 171-177.
2. J. Huh, D. Burger, and S.W. Keckler, "Exploring the Design Space of Future CMPs," *Proc. 10th Int'l Conf. Parallel Architectures and Compilation Techniques*, IEEE CS Press, 2001, pp. 199-210.
3. V. Agarwal et al., "Clock Rate vs. IPC: The End of the Road for Conventional Microprocessors," *Proc. 27th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, 2000, pp. 248-259.
4. D. Matzke, "Will Physical Scalability Sabotage Performance Gains?" *Computer*, vol. 30, no. 9, Sept. 1997, pp. 37-39.
5. P. Shivakumar and N. Jouppi, *Cacti 3.0: An Integrated Cache Timing, Power, and Area Model*, tech. report, Compaq Computer Corp., Aug. 2001.
6. R. Desikan et al., *Sim-Alpha: A Validated Execution-Driven Alpha 21264 Simulator*, tech. report TR-01-23, Dept. of Computer Sciences, Univ. Texas at Austin, 2001.
7. *Int'l Technology Roadmap for Semiconductors*, Semiconductor Industry Assoc., 2001; <http://public.itrs.net/HomeStart.htm>.
8. C. Kim, D. Burger, and S.W. Keckler, "An Adaptive, Non-Uniform Cache Structure for Wire-Delay Dominated On-Chip Caches," *Proc. Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS-10)*, ACM Press, 2002, pp. 211-222.
9. R. Kessler et al., "Inexpensive Implementations of Set-Associativity," *Proc. 16th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, 1989, pp. 131-139.
10. E. Hallnor and S. Reinhardt, "A Fully Associative Software-Managed Cache Design," *Proc. 27th Int'l Symp. Computer Architecture*, IEEE CS Press, 2000, pp. 107-116.
11. H. Pilo et al., "An 833MHz 1.5w 18Mb CMOS SRAM with 1.67Gb/s/pin," *Proc. 2000 IEEE Int'l Solid-State Circuits Conf.*, IEEE Press, 2000, pp. 266-267.

The biographies of **Changkyu Kim**, **Doug Burger**, and **Stephen W. Keckler** appear on p. 51.

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](mailto:dburger@cs.utexas.edu).

SET  
INDUSTRY  
STANDARDS

Posix  
gigabit Ethernet  
enhanced parallel ports  
wireless *token rings*  
networks **FireWire**

Computer Society members work together to define standards like  
IEEE 1003, 1394, 802, 1284, and many more.

HELP SHAPE FUTURE TECHNOLOGIES • JOIN A COMPUTER SOCIETY STANDARDS WORKING GROUP AT

**[computer.org/standards/](http://computer.org/standards/)**