Architecting Efficient Interconnects for Large Caches with CACTI 6.0

Interconnects play an increasingly important role in determining the power and performance characteristics of modern processors. An enhanced version of the popular CACTI tool primarily focuses on interconnect design for large scalable caches. The new version can help evaluate novel interconnection networks for cache access and accurately estimate the delay, power, and area of large caches with uniform and nonuniform access times.

Naveen Muralimanohar
Rajeev Balasubramonian
University of Utah

Norman P. Jouppi
Hewlett-Packard Laboratories

Efficiently executing multithreaded applications on future multicores will require fast intercore communication. Most of this communication happens via reads and writes to large shared caches in the memory hierarchy. Microprocessor performance and power will be strongly influenced by the long interconnects that must be traversed to access a cache bank. The coming decade will likely see many innovations to the multicore cache hierarchy: policies for data placement and migration, logical and physical cache reconfiguration, optimizations to the on-chip network fabric, and so on.

A large multimegabyte cache, shared by multiple threads, will likely be partitioned into many banks, connected by an interconnect fabric. The cache is referred to as a nonuniform cache architecture (NUCA) if the latency for an access is a function of the variable distance traversed on the fabric. Our analysis[1,2] shows that the contribution of the fabric to overall large cache access time and power is 30 to 60 percent, and every additional cycle in average L2 cache access time can worsen performance by approximately 1 percent. Most NUCA evaluations to date have made simple assumptions when modeling the cache's parameters; it's typically assumed that the cache is connected by a grid network such that every network hop consumes one cycle. By modeling network parameters in detail and carrying out a comprehensive design space exploration, our research shows that an optimal NUCA cache organization has remarkably different layout, performance, and power than prior studies have assumed.

Most cache evaluations today rely on the cache access modeling tool CACTI (cache access and cycle time information).[3] Although CACTI accurately models small and medium-sized caches, it has many inadequacies when modeling multimegabyte caches. Creating a tool that accurately models the properties of large caches and provides the flexibility to model new ideas

0272-1732//$20.00 © 2008 IEEE Published by the IEEE Computer Society

Significance of CACTI 6.0

CACTI versions 1 through 5 assumed a uniform cache access (UCA) model, in which a large cache is partitioned into a number of banks and connected by an H-tree network that offers uniform latency for every subarray. Even though the interbank network contributes 30 to 60 percent of large cache delay and power, CACTI was restricted to the use of this single network model and a single wire type (global wires with optimal repeater placement).

CACTI 6.0 is a significantly enhanced version of the tool that primarily focuses on interconnect design for large scalable caches. The tool includes two major extensions over earlier versions: the ability to model scalable NUCA caches and a drastically improved search space with the ability to model different types of wires and routers. The salient enhancements in CACTI 6.0 include

- incorporation of many different wire models for the interbank network (local/intermediate/global wires, repeater sizing and spacing for optimal delay or power, and low-swing differential wires);
- incorporation of models for router components (buffers, crossbar, and arbiter);
- introduction of grid topologies for nonuniform cache architecture (NUCA) and a shared bus architecture for UCA with low-swing wires;
- an algorithm for design-space exploration that models different grid layouts and estimates average bank and network latency—the design space exploration also considers different wire and router types;
- introduction of empirical network contention models to estimate the impact of network configuration, bank cycle time, and workload on average cache access delay; and
- a validation analysis of all new circuit models: low-swing differential wires and distributed resistance-capacitance (RC) model for wordlines and bitlines within cache banks (router components have been validated elsewhere).

In addition to these enhancements to the cache model, CACTI 6.0 also provides an improved user interface that enables trade-off analysis for latency, power, and area.

and trade-offs can strengthen future research evaluations.

With these challenges in mind, we've created such a tool, CACTI 6.0, that we hope will greatly improve future evaluations by introducing more rigor in the estimation of cache properties. Because strict power budgets constrain all processors today, CACTI 6.0 is more power-aware: It considers several low-power circuit components and, with an improved interface, helps researchers make latency-power trade-offs. Not only will researchers employ valid baseline architectures, but the tool's analysis will steer them toward bottlenecks and stimulate innovation. To demonstrate CACTI 6.0's features, we architected an on-chip network fabric that takes advantage of different wiring options the tool provides to improve cache access latency.

NUCA and UCA models in CACTI 6.0

Traditional uniform cache access (UCA) models suffer from severe scalability issues primarily due to the fixed worst-case latency assumed for all cache banks. A more scalable approach for future large caches is to replace the H-tree bus in a UCA with a packet-switched on-chip grid network. Then the access latency is determined by the delay to route the request and response between the bank that contains the data and the cache controller. This results in the NUCA model,[4] which has been the subject of many architectural evaluations.

CACTI 6.0 incorporates network components for NUCA models and shows that a combined design space exploration over cache and network parameters yields power- and performance-optimal points that are different from those assumed in prior studies. (See the sidebar for details on how CACTI 6.0 differs from previous versions.)

NUCA design space exploration

The CACTI 6.0 tool performs an exhaustive search. It first iterates over a number of bank organizations: the cache is partitioned into 2^N banks (where N varies from 1 to 12); for each N, the banks are organized in a grid with 2^M rows (where M varies from 0 to N). For each bank organization, CACTI 5.0 determines the optimal subarray partitioning for the cache within each bank. Each bank is associated with a router. The average delay for a cache access is computed by estimating the number of network hops to each bank, the wire delay encountered on each hop, and the cache access delay within each bank. We further assume that each traversal through a router takes R cycles, where R is a user-specified input. We can design router pipelines in many ways: A four-stage pipeline is a common approach,[5] but researchers have recently proposed speculative pipelines that take up three, two, and one pipeline stages.[5–7] Although we give the user the option to pick an aggressive or conservative router, the CACTI 6.0 tool defaults to employing a moderately aggressive router pipeline with three stages.

In the NUCA model, more partitions lead to smaller delays (and power) within

each bank but greater delays (and power) on the network because of the constant overheads associated with each router and decoder. Hence, this design space exploration is required to estimate the cache partition that yields optimal delay or power. Although the algorithm is guaranteed to find the cache structure with the lowest possible delay or power, the cache organization might not offer sufficient bandwidth to handle all the requests emerging from a multicore processor.

To address this problem, CACTI 6.0 also models contention in the network in detail. This contention model itself has two major components. If the cache is partitioned into many banks, there are more routers and links on the network, and the probability of two packets conflicting at a router decreases. Thus, a many-banked cache is more capable of meeting the bandwidth demands of a many-core system. Furthermore, we can't easily pipeline certain aspects of the cache access within a bank. The longest such delay within the cache access (typically, the bitline and sense-amp delays) represents the bank's cycle time, which is the minimum delay between successive accesses to that bank. A many-banked cache has relatively small banks and a relatively low cycle time, so it can support a higher throughput and lower wait times once a request is delivered to the bank. Both these components (lower contention at routers and banks) tend to favor a many-banked system. CACTI 6.0 includes this aspect when estimating the average access time for a given cache configuration.

To improve the NUCA model's search space, and thereby enable a user to finely tune the cache configuration, CACTI 6.0 also explores different router and wire types for the links between adjacent routers. The wires are modeled as low-swing differential wires and global wires with different repeater configurations to yield many points in the power, delay, and area spectrum. The sizes of buffers and virtual channels within a router have a major influence on router power consumption and router contention under heavy load. By varying the number of virtual channels per physical channel and the number of buffers per virtual channel, we can achieve different points on the router power-delay trade-off curve.

The contention values we use in CACTI 6.0 are empirically estimated for chip multiprocessor (CMP) models with different number of cores, cache hierarchy depth (L2 or L3), bank cycle time, and router configurations. (This methodology is described elsewhere.[2]) Figure 1a shows example contention data for multicore systems as the number of banks varies. We plan to continue augmenting the tool with empirical contention values for other relevant sets of workloads such as commercial, multithreaded, and transactional benchmarks with significant traffic from cache coherence. Users can also easily plug in their own contention data if they are dealing with workloads and architectures much different from the generic models we assume in CACTI 6.0.

Figure 1b shows an example design space exploration for a 32-Mbyte NUCA L2 cache attempting to minimize latency. The *x*-axis shows the number of banks that the cache is partitioned into. For each point on the *x*-axis, many different bank organizations are considered, and the graph shows the organization with optimal delay (averaged across all banks). The *y*-axis represents this optimal delay, and we further break it down to represent the contributing components: bank access time, link and router delay, and router and bank contention. We achieve the optimal delay when the cache is organized as a 2×4 grid of eight banks.

Improvements to the interconnect model

Power is a major problem in modern processors and will continue to be the first-order design constraint for future processors. With this trend, designing cache modules focusing singularly on performance is insufficient. Different research proposals might require different cache organizations that best match the required power and frequency budget. To address this issue, we enlarged CACTI 6.0's search space to consider design points with varying cache access power and latency values. Several interconnect choices specifically help improve the tool's search space.

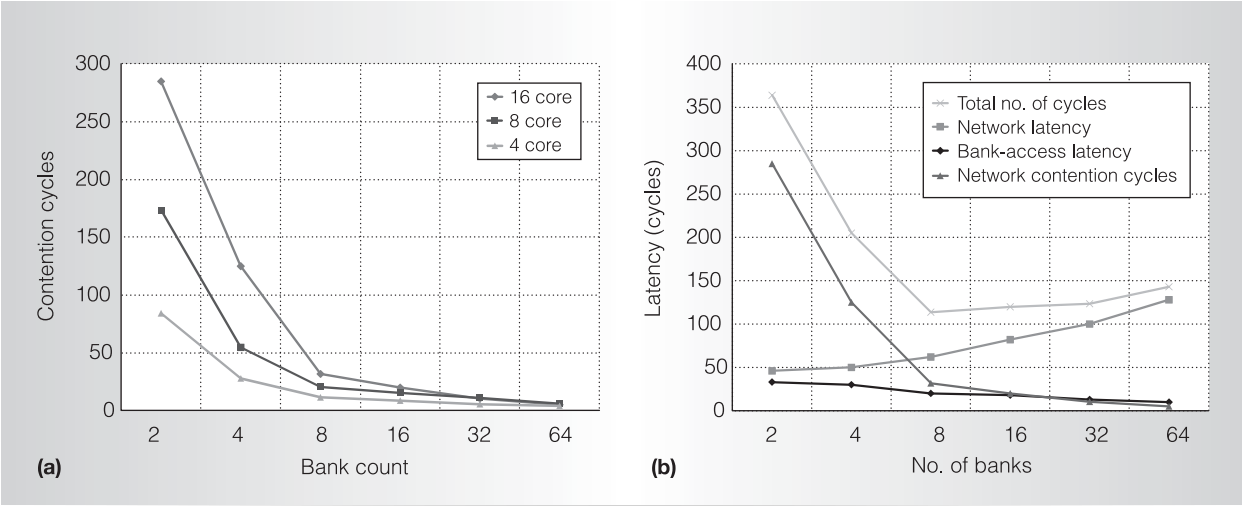

Figure 1. Nonuniform cache architecture (NUCA) design space exploration. Total network contention value/access for CMPs with different NUCA organizations (a) and optimal NUCA organization (b).

With shrinking process technologies, interconnect plays an increasingly important role in deciding the power and performance of large structures. In the deep-submicron era, the properties of a large cache are heavily impacted by the choice of the interconnect model.[1,2] Wire properties depend on numerous factors, such as wire dimensions, signaling strategy, operating voltage, operating frequency, and so on. Based on the signaling strategy, we can classify RC wires into two broad categories: traditional full-swing wires and differential, low-swing, low-power wires.

An RC wire's delay increases quadratically with its length. To avoid this quadratic relationship, a long wire is typically segmented with repeaters at regular intervals. This makes delay a linear function of wire length. However, the use of repeaters at regular intervals requires that the voltage levels of these wires swing across the full range (0 V_{DD}) for proper operation. Thus, given the quadratic dependence between voltage and power, these full-swing wires incur very high power overheads. For latency-critical transfers, we can further reduce the delay by increasing the width and spacing between the wires. Such wires have superior latency characteristics, but they have poor bandwidth for a given area. In CACTI 6.0, we include models for a latency-optimized wire that consumes eight times more area than a traditional minimum-width global wire. In addition to these two wire models, the tool also considers three more wire models that employ a small and reduced number of repeaters and hence incur 10, 20, and 30 percent delay penalties. For the sake of our discussion here, we refer to the minimum-width global wire as a B wire, the latency-optimized fat wire as an L wire, and the power optimized wires as PW wires.

One of the primary reasons for the high power dissipation of global wires is the full-swing requirement the repeaters impose. Although we can somewhat reduce the power requirement by reducing repeater size and increasing repeater spacing, the requirement is still relatively high. Low-voltage swing alternatives represent another mechanism to vary the wire power, delay, and area trade-off. Reducing the voltage swing on global wires can result in a linear reduction in power. In addition, assuming a separate voltage source for low-swing drivers will result in a quadratic savings in power. However, many caveats accompany these lucrative power savings. Because we can no longer use repeaters or latches, the delay of a low-swing wire increases quadratically with length. Because we can't pipeline such wires, they also suffer from lower throughput. A low-swing wire requires special transmitter and receiver circuits for signal

generation and amplification. This not only increases the area requirement per bit, but it also assigns a fixed cost in terms of both delay and power for each bit traversal. In spite of these issues, the power savings possible through low-swing signaling make it an attractive design choice. (The detailed methodology for the design of low-swing wires and their overhead is described elsewhere.[2])

We can also use low-swing wires within moderately sized UCA organizations. CACTI's prior versions model a large cache by partitioning it into multiple subarrays, connected with an H-tree network. This enables uniform access times for each bank and simplifies the pipelining of requests across the network. Because we can't pipeline low-swing wires and they better amortize the transmitter and receiver overhead over long transfers, instead of the H-tree network, we adopt a collection of simple broadcast buses that span all the banks (each bus spans half the banks in a column of subarrays).

As an example of the power-delay tradeoffs made possible by CACTI, we can demonstrate a design space exploration for the 16-Mbyte UCA cache in Figure 2. Figure 2a shows a power-delay curve in which each point graphed represents one of the many hundreds of cache organizations CACTI 5.0 considers while employing a single wire type (global wire with delay-optimal repeaters). The light gray points in Figure 2b represent cache organizations with power-optimized global wires considered by CACTI 6.0. The black points in Figure 2b represent the cache organizations that use low-swing wires.

This detailed design space exploration reveals cache organizations that can consume three times less power while incurring a 25 percent delay penalty.

Leveraging interconnect choices for performance optimizations

Most of the CACTI tool extensions we've discussed so far pertain to detailed interconnect models that represent the dominant contributors to cache access delay and power. Because interconnects are the primary bottleneck, we believe that researchers will actively study interconnect optimizations in the coming years. CACTI 6.0 facilitates such research by making the interconnect's contribution more explicit and incorporating various interconnect models. Using a case study of the tool's features, we can show how an architect can improve the design of the interconnect fabric employed for cache access.

Specifically, we present several techniques that leverage the special characteristics of fast L wires to improve overall processor performance. Delays within the router represent another major component of cache access time. Although we don't propose any changes to a router's microarchitecture, one of our proposals attempts to reduce the number of routers and hence its effect on access time.

Consistent with most modern implementations, we assume that each cache bank stores the tag and data arrays and that all the ways of a set are stored in a single cache bank. For most of the discussion, we also assume that there's enough metal area to support a baseline interbank network that accommodates 256 data wires and 64 address wires, all implemented as minimum-width wires on the 8X metal plane.

Early lookup

We can leverage L wires for low latency, but they consume eight times the area of a B wire on the 8X metal plane. A 16-bit L-network implementation will require that 128 B wires be eliminated to maintain constant metal area. Consider the following heterogeneous network that has the same metal area as the baseline: 128 B wires for the data network, 64 B wires for the address network, and 16 additional L wires.

In a typical cache implementation, the cache controller sends the complete address as a single message to the cache bank. After the message reaches the cache bank, it starts the lookup and selects the appropriate set. The tags of each block in the set are compared against the requested address to identify the single block that is returned to the cache controller. The address's least significant bits (LSB) are on the critical path because they are required to index into the cache bank and select candidate blocks. The

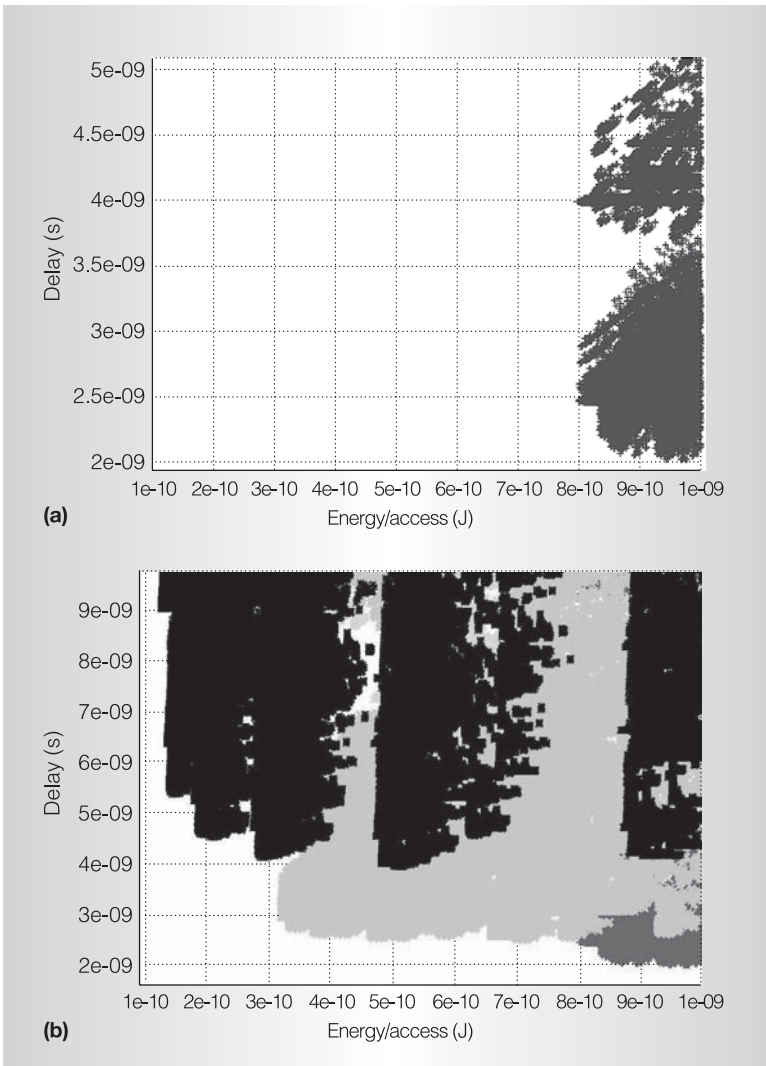

Figure 2. The power-delay trade-off in a 16-Mbyte UCA cache. Design space exploration with global wires (a) and with full-swing global wires (dark gray), wires with 30 percent delay penalty (light gray), and differential low-swing wires (black) (b).

most significant bits (MSB) are less critical because they are required only at the tag comparison stage that happens later. We can exploit this opportunity to break the traditional sequential access. A partial address consisting of LSB can be transmitted on the low-bandwidth L network, and cache access can be initiated as soon as these bits arrive at the destination cache bank. In parallel with the bank access, the block's entire address is transmitted on the slower address network composed of B wires (we refer to this design choice as option A). When the entire address arrives at the bank and when the set has been read out of the cache, the MSB is used to select at most a single cache block among the candidate blocks. The data block is then returned to the cache controller on the 128-bit-wide data network.

The proposed optimization is targeted only for cache reads. Cache writes aren't done speculatively and wait for the complete address to update the cache line.

Aggressive lookup

Although the previous proposal is effective in hiding a major part of the cache access time, it still suffers from long network delays when transmitting the entire address over the B-wire network. In an alternative implementation (option B), we can eliminate the 64-bit address network and send the entire address in a pipelined manner over the 16-bit L network. Four flits are used to transmit the address, with the first flit containing the index bits and initiating the early-lookup process. This approach increases contention in the address network and yields little performance benefit.

To reduce the contention in the L network, we introduce an optimization that we refer to as aggressive lookup (option C). By eliminating the 64-bit address network, we can increase the L network's width by eight bits without exceeding the metal area budget. Thus, in a single flit on the L network, we can transmit the index bits required for an early lookup and eight bits of the tag. For cache reads, the rest of the tag isn't transmitted on the network. This subset of the tag is used to implement a partial tag comparison at the cache bank. Cache writes still require the complete address and the address is sent in multiple flits over the L network. According to our simulations, for 99 percent of all cache reads, the partial tag comparison yields a single correct matching data block. In the remaining cases, false positives are also flagged. All blocks that flag a partial tag match must now be transmitted back to the CPU cache controller (along with their tags) to implement a full tag comparison and locate the required data. Thus, we reduce

the bandwidth demands on the address network at the cost of higher bandwidth demands on the data network. Our results show this is a worthwhile trade-off.

Clearly, depending on the application's bandwidth needs and the available metal area, any one of these three design options might perform best. The point here is that the choice of interconnect can have a major impact on cache access times and is an important consideration in determining an optimal cache organization.

Hybrid network

The delay-optimal cache organization selected by CACTI 6.0 often employs global B wires for data and address transfers. Our previous discussion makes the case that different types of wires in the address and data networks can improve performance. Hence, the CACTI tool facilitates architectural innovation and exposes avenues for additional optimizations within the network architecture. If we use fat L wires for the address network, it often takes less than a cycle to transmit a signal between routers. Therefore, part of the cycle time is wasted and most of the address network delay is attributed to router delay. Hence, we propose an alternative topology for the address network. By using fewer routers, we can take full advantage of the low-latency L network and lower the overhead from routing delays. The corresponding penalty is that the network supports a lower overall bandwidth.

Figure 3 shows the proposed hybrid topology to reduce the routing overhead in the address network for uniprocessor models. The address network is now a combination of point-to-point and bus architectures. Each row of cache banks is allocated a single router, and these routers are connected to the cache controllers with a point-to-point network composed of L wires. The cache banks in a row share a bus composed of L wires. When a cache controller receives a request from the CPU, the address is first transmitted on the point-to-point network to the appropriate row and then broadcast on the bus to all the cache banks in the row. Each hop on the point-to-point network takes a single cycle (for the 4 × 4-bank model) of link latency and three cycles of router latency. The broadcast on the bus doesn't suffer from router delays and is only a function of link latency (two cycles for the 4 × 4-bank model). Because the bus has a single master (the router on that row), there are no arbitration delays involved. If the bus latency is more than a cycle, the bus can be pipelined.[8] For the simulations in this study, we assume that the address network is always 24 bits wide (as in option C) and the aggressive-lookup policy is adopted—that is, blocks with partial tag matches are sent to the CPU cache controller. As before, the data network continues to employ the grid-based topology and links composed of B wires (128-bit network, just as in option C).

A grid-based address network (especially one composed of L wires) suffers from huge metal area and router overheads. Using a bus composed of L wires helps eliminate the metal area and router overhead, but it causes an inordinate amount of contention for this shared resource. The hybrid topology that employs multiple buses connected with a point-to-point network strikes a good balance between latency and bandwidth because multiple addresses can simultaneously be serviced on different rows. Thus, in this proposed hybrid model, we introduce three forms of heterogeneity: different types of wires in data and address networks, different topologies for data and address networks, and different architectures (bus-based and point-to-point) in different parts of the address network.

Evaluation

We evaluated the proposed models in a single-core processor and an eight-core CMP. Both uniprocessor and CMP employ a 32-Mbyte L2 cache, with the L2 shared among all cores in the CMP model. (See related work for detailed evaluation methodologies and workload choices.[1,2])

Table 1 summarizes the behavior of processor models with eight different cache configurations. The first six models help demonstrate the improvements from our most promising novel designs, and the last two models show results for other design

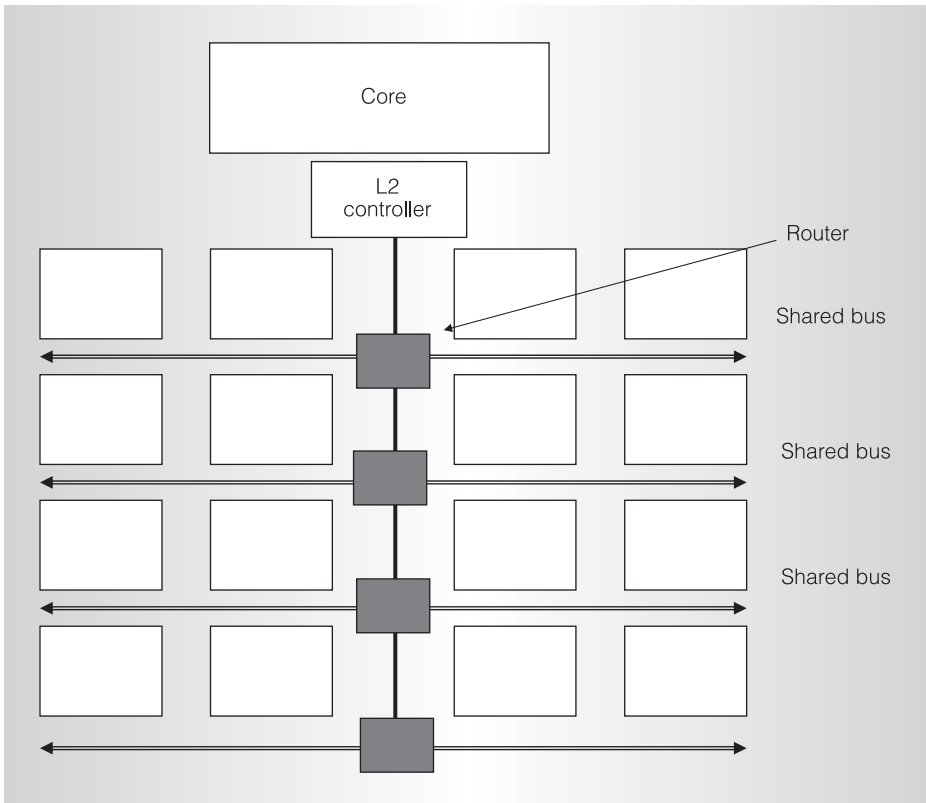

Figure 3. Hybrid network topology for a uniprocessor.

options that we considered as useful comparison points.

Model 1 is based on simpler methodologies in prior work,[4] where the bank size is calculated such that the link delay across a bank is less than one cycle. All other models employ the CACTI 6.0 tool to calculate the optimum bank count, bank access latency, and link latencies (vertical and horizontal) for the grid network. Model 2 is the baseline cache organization obtained with CACTI 6.0 that uses minimum-width wires on the 8X metal plane for the address and data links. Models 3 and 4 augment the baseline interconnect with an L network to accelerate cache access. Model 3 implements the early-lookup proposal, and model 4 implements the aggressive-lookup proposal. Model 5 simulates the hybrid network, using a combination of bus and point-to-point network for address communication. As we already discussed, the bandwidths of the links in all these simulated models are adjusted so that the net metal area is constant. All these optimizations help speed up the address network but don't attempt to improve the data network. To get an idea of the best performance possible with such optimizations to the address network, we simulated an optimistic model (model 6), where the request carrying the address magically reaches the appropriate bank in one cycle. The data transmission back to the cache controller happens on B wires just as in the other models. Models 7 and 8 are the additional organizations we're using for reference (more details are available elsewhere[1]).

Figure 4 shows the instructions per cycle (IPCs) (average across the SPEC2k suite) for all eight processor models, normalized against model 1. It also shows the average across programs in SPEC2k that are sensitive to L2 cache latency. In spite of having the least-possible bank access latency (three cycles as opposed to 17 cycles for other models), Model 1 has the poorest performance due to the high network overhead associated with each L2 access. Model 2, the optimal performance cache

Table 1. Summary of different models simulated, assuming global 8X wires for the interbank links.

Model	Network link contents count*	Description
1	B wires (256D, 64A)	Based on prior work
2	B wires (256D, 64A)	Derived from CACTI 6.0
3	B wires (128D, 64A) and L wires (16A)	Implements early lookup
4	B wires (128D) and L wires (24A)	Implements aggressive lookup
5	L wires (24A) and B wires (128D)	Latency-bandwidth trade-off
6	B wires (256D), 1-cycle Add	Implements optimistic case
7	L wires (40A/D)	Latency optimized
8	b-wires (128D) and L wires (24A)	Address-L wires and data-B wires

* A and D denote the address and data networks, respectively.

organization derived from CACTI 6.0, performs significantly better than model 1. On an average, model 2's performance is 73 percent better across all the benchmarks and 114 percent better for benchmarks that are sensitive to L2 latency. This performance improvement is accompanied by reduced power and area because it uses fewer routers.

The early-lookup optimization improves on model 2's performance. On average, model 3's performance is 6 percent better, compared to model 2 across all the benchmarks and 8 percent better for L2-sensitive benchmarks. Model 4 further improves the cache's access time by performing the early lookup and aggressively sending all the blocks that exhibit partial tag matches. This mechanism has 7 percent higher performance, compared to model 2 across all the benchmarks, and 9 percent for L2-sensitive benchmarks. The low performance improvement of model 4 is mainly due to the high router overhead associated with each transfer. The increase in data network traffic from partial tag matches is less than 1 percent.

The aggressive- and early-lookup mechanisms trade off data network bandwidth for a low-latency address network. By halving the data network's bandwidth, the delay for the pipelined transfer of a cache line increases by two cycles (because the cache line takes up two flits in the baseline data network). This enables a low-latency address network that can save two cycles on every hop, resulting in a net win in terms of overall cache access latency. The narrower data network is also susceptible to more contention cycles, but this wasn't a major factor for the evaluated processor models and workloads.

The hybrid model overcomes the shortcomings of model 4 by reducing the number of routers in the network. Aggressive lookup implemented in a hybrid topology (model 5) performs the best and is within a few percent of the optimistic model 6. Compared to model 2, the hybrid model performs 15 percent better across all benchmarks and 20 percent better for L2-sensitive benchmarks.

Figure 5 shows the IPC improvement of different models in a CMP environment.

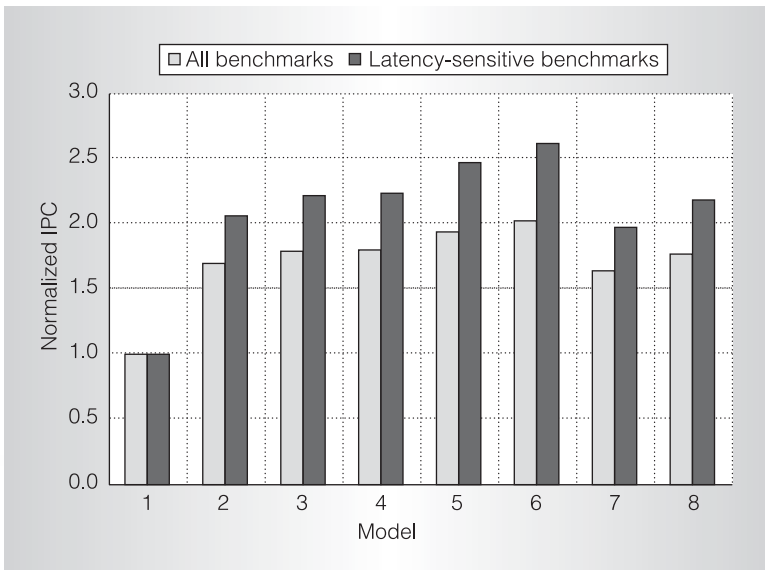

Figure 4. Normalized instructions per cycle for different L2 cache configurations on SPEC2000 benchmarks (B wires implemented on the 8X metal plane).

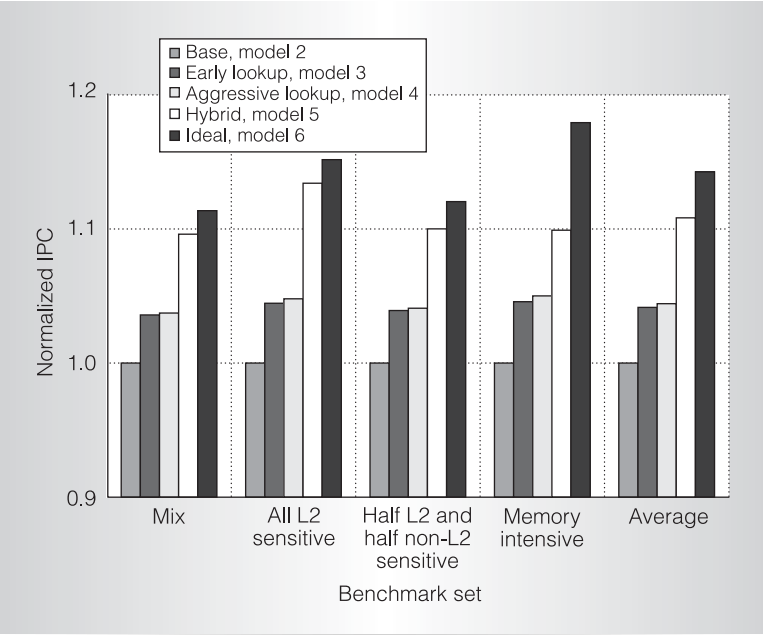

Figure 5. IPC improvement of different cache configurations in an eight-core CMP. Benchmark compositions include mix (ammp, applu, lucas, bzip2, crafty, mgrid, equake, gcc), all L2 sensitive (ammp, apsi, art, bzip2, crafty, eon, equake, gcc), half L2 and half non-L2 sensitive (ammp, applu, lucas, bzip2, crafty, mgrid, mesa, gcc), and memory intensive (applu, fma3d, art, swim, lucas, equake, gap, vpr).

We evaluate all our models for four different sets of multiprogrammed workloads (see the figure caption). Similar to our uniprocessor results, model 1 incurs a severe performance penalty due to high network overhead. Model 2, derived from CACTI 6.0, outperforms model 1 by 51 percent. Models 3, 4, and 5 yield performance improvements of 4.2 percent (early lookup), 4.5 percent (aggressive lookup), and 10.8 percent (hybrid network), respectively, over model 2. These results demonstrate the significance of interconnect design on overall processor performance. Such an evaluation is greatly simplified by having access to wire and router models in CACTI 6.0 and by having a tool that automatically generates an optimal baseline organization and exposes where the processor is spending time and energy.

We believe several enhancements can be made to CACTI 6.0 to improve research methodologies for large caches. This includes incorporating 3D models, parameter variation, and novel cell designs. Although we expect to continue adding empirical contention models for the network, we hope to eventually formulate techniques to analytically estimate network contention and consider nongrid topologies. MICRO

Acknowledgments

This work was supported in part by US National Science Foundation grant CCF-0430063 and NSF CAREER award CCF-0545959.

References

1. N. Muralimanohar and R. Balasubramonian, ''Interconnect Design Considerations for Large NUCA Caches,'' *Proc. 34th Int'l Symp. Computer Architecture* (ISCA 07), ACM Press, 2007, pp. 369-380.
2. N. Muralimanohar, R. Balasubramonian, and N.P. Jouppi, ''Optimizing NUCA Organizations and Wiring Alternatives for Large Caches With CACTI 6.0,'' *Proc. 40th Ann. IEEE/ACM Int'l Symp. Microarchitecture* (MICRO 07), IEEE CS Press, 2007, pp. 3-14.
3. S. Thoziyoor, N. Muralimanohar, and N. Jouppi, ''CACTI 5.0,'' tech. report HPL-2007-167, HP Lab., 2007.
4. C. Kim, D. Burger, and S. Keckler, ''An Adaptive, Non-Uniform Cache Structure for Wire-Dominated On-Chip Caches,'' *Proc. 10th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (ASPLOS 02), ACM Press, 2002, pp. 211-222.
5. W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2003.
6. R. Mullins, A. West, and S. Moore, ''Low-Latency Virtual-Channel Routers for On-Chip Networks,'' *Proc. 31st Int'l Symp. Computer Architecture* (ISCA 04), IEEE CS Press, 2004, p. 188.
7. L.-S. Peh and W. Dally, ''A Delay Model and Speculative Architecture for Pipelined Routers,'' *Proc. 7th IEEE Symp. High-Performance Computer Architecture* (HPCA 01), IEEE CS Press, 2001, p. 255.
8. R. Kumar, V. Zyuban, and D. Tullsen, ''Interconnections in Multi-Core Architec-

tures: Understanding Mechanisms, Overheads, and Scaling,'' *Proc. 32nd Ann. Int'l Symp. Computer Architecture* (ISCA 05), IEEE CS Press, 2005, pp. 408-419.

Naveen Muralimanohar is pursuing a PhD in computer science at the University of Utah. His research focuses on interconnect design for future communication-bound processors and large cache hierarchies. Muralimanohar has a BS in electrical and electronics engineering from the University of Madras, India.

Rajeev Balasubramonian is an assistant professor in the School of Computing at the University of Utah. His research interests include the design of high-performance microprocessors that can efficiently tolerate long on-chip wire delays, high power densities, and frequent errors. Balasubramonian has a PhD in computer science from the University of Rochester. He is a member of the IEEE.

Norm Jouppi is a fellow and director of HP's Advanced Architecture Lab. His research interests include computer memory systems, networking for cluster computing, blade system architectures, graphics accelerators, and video, audio, and physical telepresence. Jouppi has a PhD in electrical engineering from Stanford University. He is a Fellow of the IEEE and ACM.

Direct questions and comments about this article to Rajeev Balasubramonian, 50 S. Central Campus Dr., Rm. 3190, Salt Lake City, Utah 84112; rajeev@cs.utah.edu.