

# **Interconnect Design Considerations for Large NUCA Caches**

Naveen Muralimanohar, Rajeev Balasubramonian

School of Computing, University of Utah

{naveen, rajeev}@cs.utah.edu

## **Abstract**

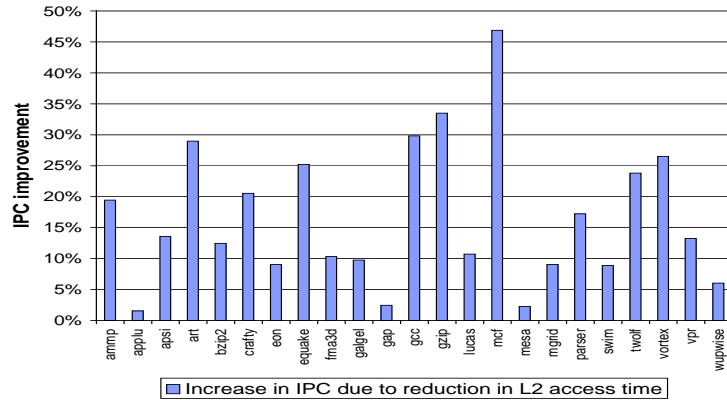
*The ever increasing sizes of on-chip caches and the growing domination of wire delay necessitate significant changes to cache hierarchy design methodologies. Many recent proposals advocate splitting the cache into a large number of banks and employing a network-on-chip (NoC) to allow fast access to nearby banks (referred to as Non-Uniform Cache Architectures (NUCA)). Most studies on NUCA organizations have assumed a generic NoC and focused on logical policies for cache block placement, movement, and search. Since wire and router delay are major limiting factors in modern processors, this work focuses on interconnect design and its influence on NUCA performance and power. We extend the widely-used CACTI cache modeling tool to take network design parameters into account. With these overheads appropriately accounted for, the optimal cache organization is typically very different from that assumed in prior NUCA studies. To alleviate the interconnect delay bottleneck, we propose novel cache access optimizations that introduce heterogeneity within the inter-bank network. The careful consideration of interconnect choices for a large cache results in a 53% performance improvement over a baseline generic NoC and the introduction of heterogeneity within the network yields an additional 11-14% performance improvement.*

**Keywords:** *cache models, non-uniform cache architectures (NUCA), memory hierarchies, on-chip interconnects, data prefetch.*

## **1. Introduction**

The shrinking of process technologies enables many cores and large caches to be incorporated into future chips. The Intel Montecito processor accommodates two Itanium cores and two private 12 MB L3 caches [24]. Thus, more than 1.2 billion of the Montecito's 1.7 billion transistors are dedicated for cache hierarchies. Every new generation of processors will likely increase the number of cores and the sizes of the on-chip cache space. If 3D technologies become a reality, entire dies may be dedicated for the implementation of a large cache [21].

Large multi-megabyte on-chip caches require global wires carrying signals across many milli-meters. It is well known that while arithmetic computation continues to consume fewer pico-seconds and die area with every



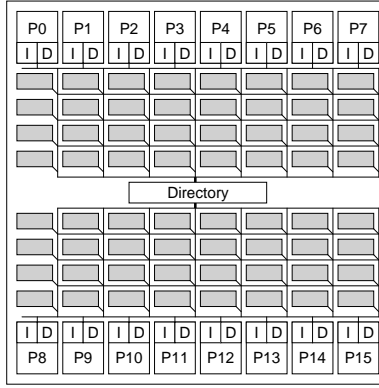
**Figure 1. Improvement in IPC due to reduction in L2 access time from 30 cycles to 15 cycles on an aggressive out-of-order processor model.**

generation, the cost of on-chip communication continues to increase [23]. Electrical interconnects are viewed as a major limiting factor, with regard to latency, bandwidth, and power. The ITRS roadmap projects that global wire speeds will degrade substantially at smaller technologies and a signal on a global wire can consume over 12 ns (60 cycles at a 5 GHz frequency) to traverse 20 mm at 32 nm technology [29]. In some Intel chips, half the total dynamic power is attributed to interconnects [22].

To understand the impact of L2 cache access times on overall performance, Figure 1 shows IPC improvements for SPEC2k programs when the L2 access time is reduced from 30 to 15 cycles (simulation methodologies are discussed in Section 4). Substantial IPC improvements (17% on average) are possible in many programs if we can reduce L2 cache access time by 15 cycles. Since L2 cache access time is dominated by interconnect delay, this paper focuses on efficient interconnect design for the L2 cache.

Caches are typically partitioned into many banks and in a uniform-cache-architecture (UCA) model, the cache access latency is determined by the latency for the furthest cache bank. This is not a scalable model as cache sizes and the latency differences between the nearest and furthest cache banks grow. To address this problem, non-uniform cache architectures (NUCA) have been proposed [18]. In a NUCA cache, the banks are connected with an interconnect fabric and the access time for a block is a function of the delays experienced in traversing the network path from the cache controller to the bank that contains the block. Figure 2 shows an example of such a NUCA cache from a recent paper that partitions the L2 cache into 64 banks [17]. Recent studies have explored policies for data placement, data movement, and data search in NUCA caches [6, 11, 12, 17, 18].

For the past several years, academic researchers have relied on CACTI [33], a cache access modeling tool, to find the optimal design points for on-chip caches. CACTI is an analytical tool that takes a set of cache parameters as



**Figure 2. 16-way CMP with a 64-bank NUCA L2 cache, based on the organization in [17].**

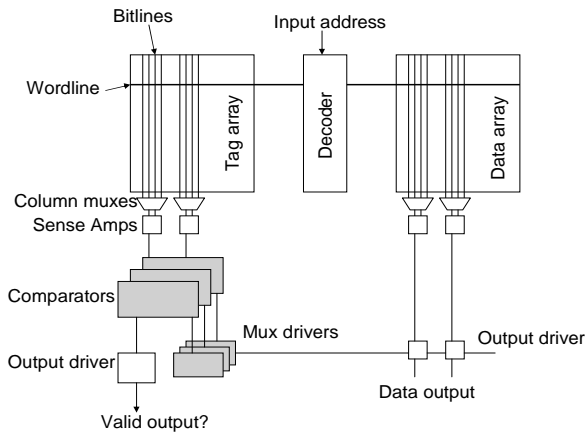
input and estimates the access time, layout, and power consumption of on-chip caches. While CACTI is powerful enough to model moderately sized UCA designs, it does not have support for NUCA designs. Most studies on NUCA caches model the inter-bank network as a grid with single-cycle hops. This is a reasonable baseline when evaluating logical cache policies. However, we make the case that interconnect parameters can have a dramatic influence on the performance and power characteristics of large NUCA caches. We extend CACTI to model interconnect properties for a NUCA cache and show that a combined design space exploration over cache and network parameters yields performance- and power-optimal cache organizations that are quite different from those assumed in prior studies. In the second half of the paper, we show that the incorporation of heterogeneity within the inter-bank network enables a number of optimizations to accelerate cache access. These optimizations can hide a significant fraction of network delay, resulting in an additional performance improvement of 14%.

Section 2 describes the CACTI cache access model and our extensions to it. Section 3 describes techniques to take advantage of heterogeneous interconnect properties and improve performance. Quantitative results are presented in Section 4. Related work is discussed in Section 5 and conclusions are drawn in Section 6.

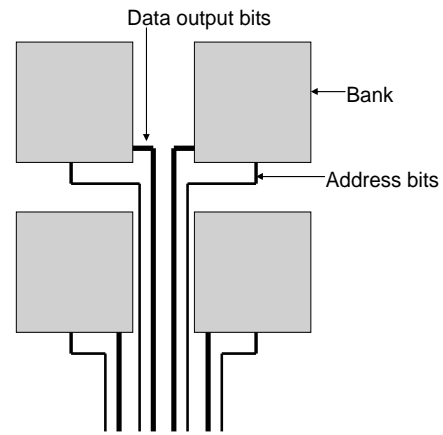
## 2. Interconnect Models for the Inter-Bank Network

### 2.1. The CACTI Model

Figure 3(a) shows the basic logical structure of a UCA cache. The address is provided as input to the decoder, which then activates a wordline in the data array and tag array. The contents of an entire row (referred to as a *set*) are placed on the bitlines, which are then sensed. The multiple tags thus read out of the tag array are compared against the input address to detect if one of the ways of the set does contain the requested data. This comparator logic drives the multiplexor that finally forwards at most one of the ways read out of the data array back to the requesting processor.



(a) Logical organization of a cache.



(b) Example physical organization of the data array.

**Figure 3. Logical and physical organization of the cache (from CACTI-3.0 [31]).**

The CACTI-4.0 cache access model [33] takes in the following major parameters as input: cache capacity, cache block size (also known as cache line size), cache associativity, technology generation, number of ports, and number of independent banks (not sharing address and data lines). As output, it produces the cache configuration that minimizes delay (with a few exceptions), along with its power and area characteristics. CACTI models the delay/power/area of eight major cache components: decoder, wordline, bitline, senseamp, comparator, multiplexor, output driver, and inter-bank wires. The wordline and bitline delays are two of the most significant components of the access time. The wordline delay is proportional to the width of each array and the bitline delay is proportional to the height of each array. In practice, the tag and data arrays are large enough that it is inefficient to implement them as single large structures. Hence, CACTI partitions each storage array (in the horizontal and vertical dimensions) to produce smaller *banks* and reduce wordline and bitline delays. Each bank has its own decoder and some central pre-decoding is now required to route the request to the correct bank. The most recent version of CACTI employs a model for semi-global (intermediate) wires and an H-tree network to compute the delay between the pre-decode circuit and the furthest cache bank. CACTI carries out an exhaustive search across different bank counts and bank aspect ratios to compute the cache organization with optimal total delay. Typically, the cache is organized into a handful of banks. An example of the cache's physical structure is shown in Figure 3(b).

## 2.2. Models for Wires and Routers

In a NUCA cache (such as the one shown in Figure 2), the size of the bank determines the number of routers and the length of the links between routers. Hence, before we extend CACTI to model a NUCA cache, we derive basic analytical models for the delay and power of wires and routers.

### Wire Models

This sub-section describes the wire model employed for the on-chip network. We begin with a quick review of factors that influence wire properties. It is well-known that the delay of a wire is a function of its RC time constant (R is resistance and C is capacitance). Resistance per unit length is (approximately) inversely proportional to the width of the wire [16]. Likewise, a fraction of the capacitance per unit length is inversely proportional to the spacing between wires, and a fraction is directly proportional to wire width. These wire properties provide an opportunity to design wires that trade off bandwidth and latency. By allocating more metal area per wire and increasing wire width and spacing, the net effect is a reduction in the RC time constant. This leads to a wire design that has favorable latency properties, but poor bandwidth properties (as fewer wires can be accommodated in a fixed metal area). Further, researchers are actively pursuing transmission line implementations that enable extremely low communication latencies [9, 14]. However, transmission lines also entail significant metal area overheads in addition to logic overheads for sending and receiving [5, 9]. If transmission line implementations become cost-effective at future technologies, they represent another attractive wire design point that can trade off bandwidth for low latency.

Similar trade-offs can be made between latency and power consumed by wires. Global wires are usually composed of multiple smaller segments that are connected with repeaters [1]. The size and spacing of repeaters influences wire delay and power consumed by the wire. When smaller and fewer repeaters are employed, wire delay increases, but power consumption is reduced. The repeater configuration that minimizes delay is typically very different from the repeater configuration that minimizes power consumption.

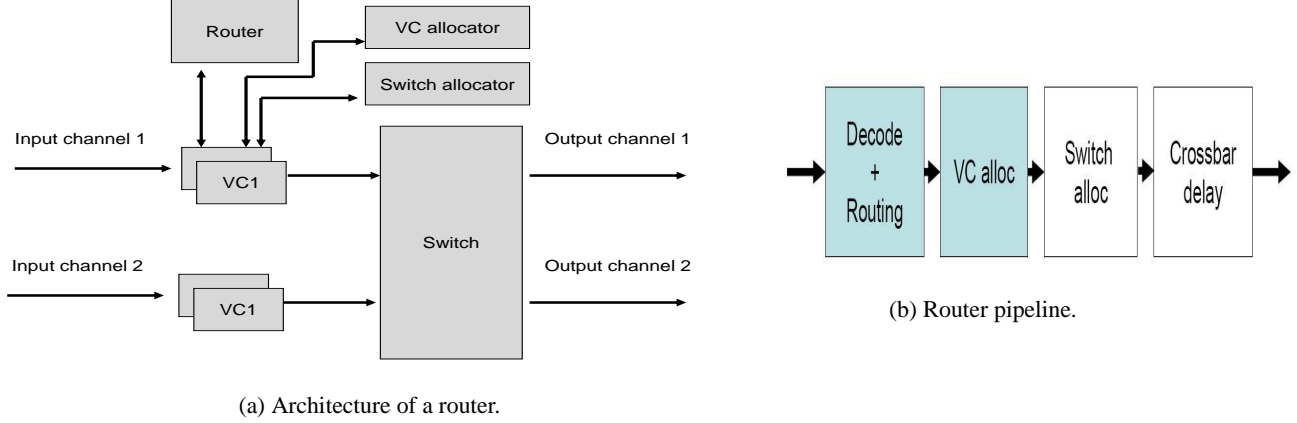
Thus, by varying properties such as wire width/spacing and repeater size/spacing, we can implement wires with different latency, bandwidth, and power properties. In this paper, we will primarily focus on three types of wires:

- *4X-B-Wires*: These are minimum-width wires on the 4X metal plane. These wires have high bandwidth and relatively high latency characteristics and are often also referred to as semi-global or intermediate wires.
- *8X-B-Wires*: These are minimum-width wires on the 8X metal plane. They are wider wires and hence have relatively low latency and low bandwidth (also referred to as global wires).
- *L-Wires*: These are *fat* wires on the 8X metal plane that consume eight times as much area as an 8X-B-wire. They offer low latency and low bandwidth.

Table 1 summarizes the relative and absolute latency and area characteristics of these wires. The calculations are based on equations in [3, 16] and ITRS roadmap parameters [29]. The use of power-optimized wires is left as future work.

Wire Type	Relative Latency	Relative Area	Latency (ns/mm)	Wiring Pitch (nm)
8X-B-Wire	1x	1x	0.122	210
4X-B-Wire	2.0x	0.5x	0.244	105
L-Wire	0.25x	8x	0.03	1680

**Table 1. Delay and area characteristics of different wire implementations at 65 nm technology.**



**Figure 4. Router architecture [15].**

## Router Models

The ubiquitous adoption of the system-on-chip (SoC) paradigm and the need for high bandwidth communication links between different modules have led to a number of interesting proposals targeting high-speed network switches/routers [13, 25, 26, 27, 28]. This section provides a brief overview of router complexity and different pipelining options available. It ends with a summary of the delay and power assumptions we make for our NUCA CACTI model. For all of our evaluations, we assume virtual channel flow control because of its high throughput and ability to avoid deadlock in the network [13].

Figure 4(a) shows a typical virtual channel router architecture and Figure 4(b) shows the different steps involved in routing a message to the appropriate destination [15]. A flit is the smallest unit of flow control and is usually the number of bits transmitted on a link in a single cycle. The size of the message sent through the network is measured in terms of flits. Every network message consists of a head flit that carries details about the destination of the message and a tail flit indicating the end of the message. If the message size is very small, the head flit can also serve the tail flit's functionality. The highlighted blocks in Figure 4(b) correspond to stages that are specific to head flits. Whenever a head flit of a new message arrives at an input port, the router stores the message in the input buffer and the input controller decodes the message to find the destination. After the decode process, it is then fed to a virtual channel (VC) allocator. The VC allocator consists of a set of arbiters and control logic that takes in

requests from messages in all the input ports and allocates appropriate output virtual channels at the destination. If two head flits compete for the same channel, then depending on the priority set in the arbiter, one of the flits gains control of the VC. Upon successful allocation of the VC, the head flit proceeds to the switch allocator. Once the decoding and VC allocation of the head flit are completed, the remaining flits perform nothing in the first two stages. The switch allocator reserves the crossbar so the flits can be forwarded to the appropriate output port. Finally, after the entire message is handled, the tail flit de-allocates the VC and clears all the decode buffers. Thus, a typical router pipeline consists of four different stages with the first two stages playing a role only for head flits.

Peh et al. [27] propose a speculative router model to reduce the pipeline depth of virtual channel routers. In their pipeline, switch allocation happens speculatively, in parallel with VC allocation. If the VC allocation is not successful, the message is prevented from entering the final stage, thereby wasting the reserved crossbar time slot. To avoid performance penalty due to mis-speculation, the switch arbitration gives priority to non-speculative requests over speculative ones. This new model implements the router as a three-stage pipeline.

The bulk of the delay in router pipeline stages comes from arbitration and other control overheads. Mullins et al. [26] remove the arbitration overhead from the critical path by pre-computing the grant signals. The arbitration logic pre-computes the grant signal based on requests in previous cycles. If there are no requests present in the previous cycle, one viable option is to speculatively grant permission to all the requests. If two conflicting requests get access to the same channel, one of the operations is aborted. While successful speculations result in a single-stage router pipeline, mis-speculations are expensive in terms of delay and power.

Single stage router pipelines are not yet a commercial reality. At the other end of the spectrum is the high speed 1.2 GHz router in the Alpha 21364 [25]. The router has eight input ports and seven output ports that includes four external ports to connect off-chip components. The router is deeply pipelined with nine pipeline stages (including special stages for wire delay and ECC) to allow the router to run at the same speed as the main core.

Essentially, innovations in router microarchitectures are on-going. The pipeline depth ranges from a single cycle speculative model [26] to a nine stage model in the Alpha 21364 [25]. For the purpose of our study, we adopt the moderately aggressive implementation with a 3-stage pipeline [27]. Our power model is also derived from a corresponding analysis by some of the same authors [34]. As a sensitivity analysis, our results also show the effect of employing router microarchitectures with different pipeline latencies.

The major contributors to router power are the crossbars, buffers, and arbiters. Our router power calculation is based on the analytical models derived by Wang et al. [34, 35]. For updating CACTI with network power values,

Component	Address Router Energy (J)	Data Router Energy (J)
Arbiter	1.7626e-13	2.1904e-13
Crossbar	9.17791e-12	1.19788e-10
Buffer	3.77034e-12	1.48893e-11
Total static energy	3.57168e-12	1.38513e-11

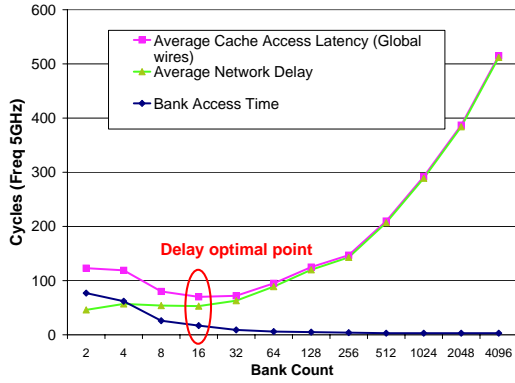
**Table 2. Energy consumed (max) by arbiters, buffers, and crossbars for a 32-byte transfer.**

we assume a separate network for address and data transfer. Each router has five input and five output ports and each physical channel has four virtual channels. Table 2 shows the energy consumed by each router at 65 nm for a 5 GHz clock frequency.

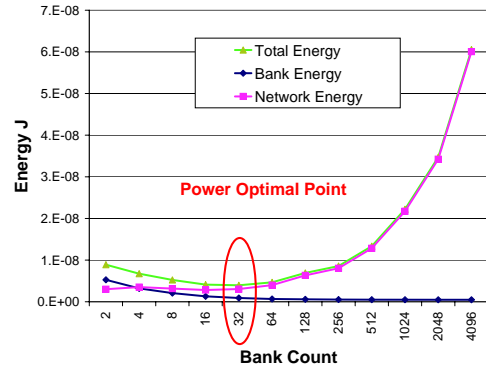
### 2.3. Extensions to CACTI

For an arbitrary NUCA cache organization, the delay for a cache request is determined by the number of links that must be traversed, the delay for each link, the number of routers that are traversed, the delay for each router, the access time within each bank, and the contention cycles experienced at the routers. For now, we will assume that the contention cycles contribute marginally and these will not be modeled analytically (they are modeled in detail in our architectural simulations in Section 4).

For a given total cache size, we partition the cache into  $2^N$  cache banks ( $N$  varies from 1 to 12) and for each  $N$ , we organize the banks in a grid with  $2^M$  rows ( $M$  varies from 0 to  $N$ ). For each of these cache organizations, we compute the average access time for a cache request as follows. The cache bank size is first fed to unmodified CACTI-4.0 to derive the delay-optimal UCA organization for that cache size. CACTI-4.0 also provides the corresponding dimensions for that cache size. The cache bank dimensions along with the router dimensions obtained from [35] enable the calculation of wire lengths between successive routers. Based on delays for B-wires (Table 1), we compute the delay for a link (and round up to the next cycle for a 5 GHz clock). The (uncontended) latency per router is assumed to be three cycles. The delay for a request is a function of the bank that services the request and if we assume a random distribution of accesses, the average latency can be computed by simply iterating over every bank, computing the latency for each bank access, and taking the average. During this design space exploration over NUCA organizations, we keep track of the cache organization that minimizes a given metric (in this study, either average latency or average power per access). These preliminary extensions to CACTI are referred to as *CACTI-L2*. We can extend the design space exploration to also include different wire types, topologies, and router configurations, and include parameters such as metal/silicon area and bandwidth in our objective function. For now, we simply show results for performance- and power-optimal organizations with various wire and router microarchitecture assumptions.



(a) Access latency for a 32 MB cache.



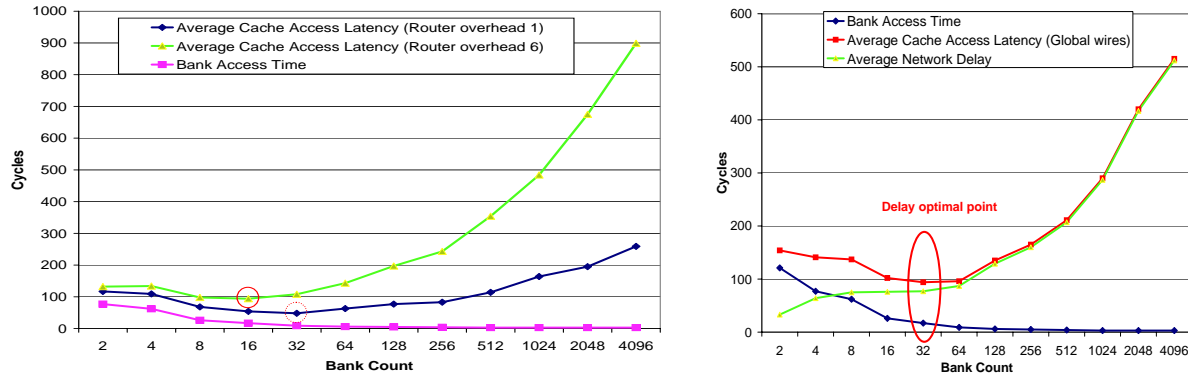
(b) Average energy consumption for a 32 MB cache.

**Figure 5. Access latency and energy for a 32 MB cache as a function of the number of banks. The contributions of the bank and network to latency and energy are also shown.**

## 2.4. CACTI-L2 Results

For a given total cache size, if the number of cache banks increases, the delay within a bank and the latency per hop on the network reduce, but the average number of network hops for a request increases (assuming a grid network). Figure 5(a) shows the effect of bank count on total average (uncontended) access time for a 32 MB NUCA cache and breaks this access time into delay within a bank and delay within the inter-bank network. We assume a grid network for inter-bank communication, global 8X-B wires for all communication, and a 3-cycle router pipeline. For each point on the curve, the bank access time is computed by feeding the corresponding bank size to the unmodified version of CACTI. The (uncontended) network delay is computed by taking the average of link and router delay to access every cache bank.

Not surprisingly, bank access time is proportional to bank size (or inversely proportional to bank count). For bank sizes smaller than 64KB (that corresponds to a bank count of 512), the bank access time is dominated by logic delays in each stage and does not vary much. The average network delay is roughly constant for small values of bank count (some noise is introduced because of discretization and from irregularities in aspect ratios). When the bank count is quadrupled, the average number of hops to reach a bank roughly doubles. But, correspondingly, the hop latency does not decrease by a factor of two because of the constant area overheads (decoders, routers, etc.) associated with each bank. Hence, for sufficiently large bank counts, the average network delay keeps increasing. The graph shows that the selection of an appropriate bank count value is important in optimizing average access time. For the 32 MB cache, the optimal organization has 16 banks, with each 2 MB bank requiring 17 cycles for the bank access time. We note that prior studies [6, 17] have sized the banks (64 KB) so that each hop on the network is a single cycle. According to our models, partitioning the 32 MB cache into 512 64KB banks would



(a) Effect of router overhead on optimum bank count for 32 MB cache.

(b) Optimum bank count value for 64MB cache.

**Figure 6. Effect of router overhead and cache size on access latency.**

result in an average access time that is more than twice the optimal access time. However, increased bank count can provide more bandwidth for a given cache size. The above analysis highlights the importance of the proposed network design space exploration in determining the optimal NUCA cache configuration. As a sensitivity analysis, we show the corresponding access time graphs for various router delays and increased cache size in Figure 6.

Similar to the analysis above, we chart the average power consumption per access as a function of the bank count in Figure 5(b). A large bank causes an increase in dynamic power when accessing the bank, but reduces the number of routers and power dissipated in the network. We evaluate different points on this trade-off curve and select the configuration that minimizes power. The bank access dynamic power is based on the output of CACTI. The total leakage power for all banks is assumed to be a constant for the entire design space exploration as the total cache size is a constant. Wire power is calculated based on ITRS data and found to be  $2.892 \cdot af + 0.6621$  (W/m),  $af$  being the activity factor and 0.6621 is the leakage power in the repeaters. We compute the average number of routers and links traversed for a cache request and use the data in Tables 1 and 2 to compute the network dynamic power.

### 3. Leveraging Interconnect Choices for Performance Optimizations

The previous section describes our modifications to CACTI and the different wire implementations possible in a network. Wires can be optimized for either high bandwidth, low latency, or low power and these choices can be leveraged to customize the L2 cache inter-bank network. Delays within the router are also a major component of cache access time, but this paper does not attempt any changes to the microarchitecture of a router (one of our proposals attempts to reduce the number of routers).

Consistent with most modern implementations, it is assumed 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 this discussion, we will assume that there

is enough metal area to support a baseline inter-bank network that accommodates 256 data wires and 64 address wires, all implemented as minimum-width wires on the 8X metal plane (the 8X-B-wires in Table 1). A higher bandwidth inter-bank network does not significantly improve IPC, so we believe this is a reasonable baseline. Next, we will consider optimizations that incorporate different types of wires, without exceeding the above metal area budget.

### 3.1. Early Look-Up

L-wires can be leveraged for low latency, but they consume eight times the area of a B-wire on the 8X metal plane. The implementation of a 16-bit L-network 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 look-up 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. Thus, all the operations happen in a sequential manner resulting in a large total access time. We observe that the least significant bits of the address (LSB) are on the critical path because they are required to index into the cache bank and select candidate blocks. The most significant bits (MSB) are less critical since 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 entire address of the block 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.

For a 512 KB cache bank with a block size of 64 bytes and a set associativity of 8, only 10 index bits are required to read a set out of the cache bank. Hence, the 16-bit L-network is wide enough to accommodate the index bits and additional control signals (such as destination bank). In terms of implementation details, the coordination between the address transfers on the L-network and the slower address network can be achieved in the following manner. We allow only a single early look-up to happen at a time and the corresponding index bits are maintained in a register. If an early look-up is initiated, the cache bank pipeline proceeds just as in the base case

until it arrives at the tag comparison stage. At this point, the pipeline is stalled until the entire address arrives on the slower address network. When this address arrives, it checks to see if the index bits match the index bits for the early look-up currently in progress. If the match is successful, the pipeline proceeds with tag comparison. If the match is unsuccessful, the early look-up is squashed and the entire address that just arrived on the slow network is used to start a new L2 access from scratch. Thus, an early look-up is wasted if a different address request arrives at a cache bank between the arrival of the LSB on the L-network and the entire address on the slower address network. If another early look-up request arrives while an early look-up is in progress, the request is simply buffered (potentially at intermediate routers). For our simulations, supporting multiple simultaneous early look-ups was not worth the complexity.

The early look-up mechanism also introduces some redundancy in the system. There is no problem if an early look-up fails for whatever reason – the entire address can always be used to look up the cache. Hence, the transmission on the L-network does not require ECC or parity bits.

Apart from the network delay component, the major contributors to the access latency of a cache are delay due to decoders, wordlines, bitlines, comparators, and drivers. Of the total access time of the cache, depending on the size of the cache bank, around 60-80% of the time has elapsed by the time the candidate sets are read out of the appropriate cache bank. By breaking the sequential access as described above, much of the latency for decoders, bitlines, wordlines, etc., is hidden behind network latency. In fact, with this optimization, it may even be possible to increase the size of a cache bank without impacting overall access time. Such an approach will help reduce the number of network routers and their corresponding power/area overheads. In an alternative approach, circuit/VLSI techniques can be used to design banks that are slower and consume less power (for example, the use of body-biasing and high-threshold transistors). The exploration of these optimizations is left for future work.

### **3.2. Aggressive Look-Up**

While the previous proposal is effective in hiding a major part of the cache access time, it still suffers from long network delays in the transmission of the entire address over the B-wire network. In an alternative implementation (referred to as *option-B*), the 64-bit address network can be eliminated and the entire address is sent 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 look-up process. In Section 4, we show that 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 look-up*

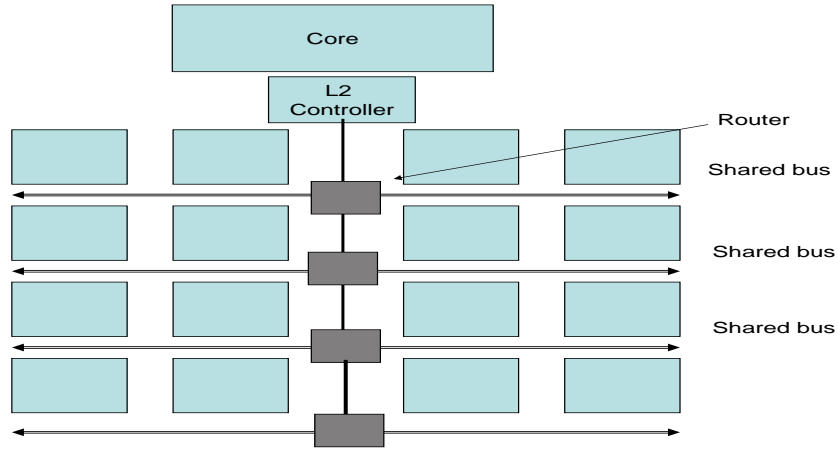
(or *option-C*). By eliminating the 64-bit address network, we can increase the width of the L-network by eight bits without exceeding the metal area budget. Thus, in a single flit on the L-network, we can not only transmit the index bits required for an early look-up, but also eight bits of the tag. The rest of the tag is not transmitted on the network. This sub-set of the tag is used to implement a partial tag comparison at the cache bank. According to our simulations, for 99% of all cache hits, 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 are reducing the bandwidth demands on the address network at the cost of higher bandwidth demands on the data network. As we show in the results, this is a worthwhile trade-off.

With the early look-up optimization, multiple early look-ups at a bank are dis-allowed to simplify the task of co-ordinating the transmissions on the L and B networks. The aggressive look-up optimization does not require this co-ordination, so multiple aggressive look-ups can proceed simultaneously at a bank. On the other hand, ECC or parity bits are now required for the L-network because there is no B-network transmission to fall back upon in case of error. In a CMP, the L-network must also include a few bits to indicate where the block must be sent to. Partial tag comparisons exhibit very good accuracy even if only five tag bits are used, so the entire address request may still fit in a single flit. The L-network need not accommodate the MSHR-id as the returned data block is accompanied with the full tag.

Clearly, depending on the bandwidth needs of the application and the available metal area, any one of the three discussed design options may 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. Given our set of assumptions, our results in the next section show that option-C performs best, followed by option-A, followed by option-B.

### **3.3. Hybrid Network**

The optimal cache organization selected by CACTI-L2 is based on the assumption that each link employs B-wires for data and address transfers. The discussion in the previous two sub-sections makes the case that different types of wires in the address and data networks can improve performance. If L-wires are employed 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 employing fewer routers, we take full advantage of the low latency L-network



**Figure 7. Hybrid network topology for a uniprocessor model**

and lower the overhead from routing delays. The corresponding penalty is that the network supports a lower overall bandwidth.

Figure 7 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 4x4-bank model) of link latency and three cycles of router latency. The broadcast on the bus does not suffer from router delays and is only a function of link latency (2 cycles for the 4x4 bank model). Since 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 [19]. For the simulations in this study, we assume that the address network is always 24 bits wide (just as in option-C above) and the aggressive look-up policy is adopted (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 above).

A grid-based address network (especially one composed of L-wires) suffers from huge metal area and router overheads. The use of a bus composed of L-wires helps eliminate the metal area and router overhead, but 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 as multiple addresses can simultaneously be serviced on different rows. Thus, in this proposed hybrid model, we have introduced three forms of heterogeneity: (i) different types of wires are being used in data and address networks, (ii) different

Fetch queue size	64	Branch predictor	comb. of bimodal and 2-level
Bimodal predictor size	16K	Level 1 predictor	16K entries, history 12
Level 2 predictor	16K entries	BTB size	16K sets, 2-way
Branch mispredict penalty	at least 12 cycles	Fetch width	8 (across up to 2 basic blocks)
Dispatch and commit width	8	Issue queue size	60 (int and fp, each)
Register file size	100 (int and fp, each)	Re-order Buffer size	80
L1 I-cache	32KB 2-way	L1 D-cache	32KB 2-way set-associative, 3 cycles, 4-way word-interleaved
L2 cache	32MB 8-way SNUCA		
L2 Block size	64B		
I and D TLB	128 entries, 8KB page size	Memory latency	300 cycles for the first chunk

**Table 3. Simplescalar simulator parameters.**

topologies are being used for data and address networks, (iii) the address network uses different architectures (bus-based and point-to-point) in different parts of the network.

## 4. Results

### 4.1. Methodology

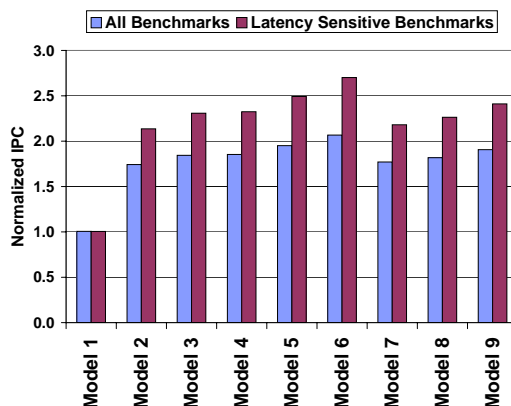
Our simulator is based on Simplescalar-3.0 [7] for the Alpha AXP ISA. Table 3 summarizes the configuration of the simulated system. All our delay and power calculations are for a 65 nm process technology and a clock frequency of 5 GHz. Contention for memory hierarchy resources (ports and buffers) is modeled in detail. We assume a 32 MB on-chip level-2 static-NUCA cache and employ a grid network for communication between different L2 banks. The network employs two unidirectional links between neighboring routers and virtual channel flow control for packet traversal. The router has five input and five output ports. We assume four virtual channels for each physical channel and each channel has four buffer entries (since the flit counts of messages are small, four buffers are enough to store an entire message). The network uses adaptive routing similar to the Alpha 21364 network architecture [25]. If there is no contention, a message attempts to reach the destination by first traversing in the horizontal direction and then in the vertical direction. If the message encounters a stall, in the next cycle, the message attempts to change direction, while still attempting to reduce the Manhattan distance to its destination. We evaluate all our proposals for uniprocessor and CMP processor models. Our CMP simulator is also based on Simplescalar and employs eight out-of-order cores and a shared 32MB level-2 cache. For most simulations, we assume the same network bandwidth parameters outlined in Section 3 and reiterated in Table 4. Since network bandwidth is a bottleneck in the CMP, we also show CMP results with twice as much bandwidth. As a workload, we employ SPEC2k programs executed for 100 million instruction windows identified by the Simpoint toolkit [30]. The composition of programs in our multi-threaded CMP workload is described in the next sub-section.

### 4.2. IPC Analysis

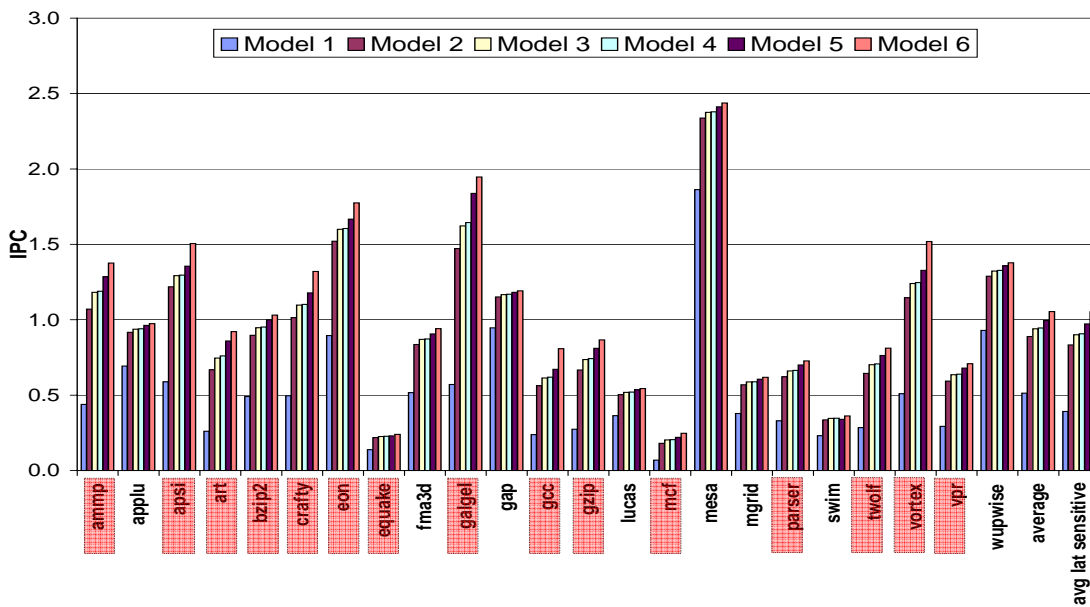
We begin by examining the behavior of processor models with eight different cache configurations (summarized in Table 4). The first six models help demonstrate the improvements from our most promising novel designs, and

Model	Hop latency (vert,horiz)	Bank access time	Bank count	Network link contents	Description
Model 1	1,1	3	512	B-wires (256D, 64A)	Based on prior work
Model 2	4,3	17	16	B-wires (256D, 64A)	Derived from CACTIL2
Model 3	4,3	17	16	B-wires (128D, 64A) & L-wires (16A)	Implements early lookup
Model 4	4,3	17	16	B-wires (128D) & L-wires (24A)	Implements aggressive lookup
Model 5	hybrid	17	16	L-wires (24A) & B-wires (128D)	Latency-bandwidth tradeoff
Model 6	4,3	17	16	B-wires (256D), 1cycle Add	Implements optimistic case
Model 7	1,1	17	16	L-wires (40A/D)	Latency optimized
Model 8	4,3	17	16	B-wires (128D) & L-wires (24A)	Address-L-wires & Data-B-wires

**Table 4.** Summary of different models simulated. Global 8X wires are assumed for the inter-bank links. “A” and “D” denote the address and data networks, respectively.



**Figure 8.** Normalized IPCs of SPEC2000 benchmarks for different L2 cache configurations (B-wires implemented on the 8X metal plane).



**Figure 9.** IPCs of SPEC2000 benchmarks for different L2 cache configurations (B-wires implemented on the 8X metal plane).

Bank Count	Bank Access Time	Average Cache Access Time	Early Lookup	Aggressive Fetch	Optimistic Model
2	77	122	105.5	105.5	100.5
4	62	119	100.5	100.5	91.5
8	26	80	65.6	65.0	54.0
16	17	70	60.5	59.5	44.5
32	9	72	66.0	64.5	41.5
64	6	86	81.3	78.0	47.0
128	5	108	105.6	104.5	57.5
256	4	147	144.4	139.5	76.5
512	3	210	207.2	202.5	107.5
1024	3	292	289.1	275.5	148.5
2048	3	387	387.0	387.0	196.0
4096	3	515	515.0	515.0	260.0

**Table 5. Access latencies for different cache configurations. The message transfers are assumed to happen in 8x wires.**

the last two models show results for other design options that were also considered and serve as useful comparison points.

The first model is based on methodologies in prior work [18], where the bank size is calculated such that the routing delay across a bank is less than one cycle. All other models employ the proposed CACTI-L2 tool to calculate the optimum bank count, bank access latency, and hop latencies (vertical and horizontal) for the grid network. Model two is the baseline cache organization obtained with CACTI-L2 that employs minimum-width wires on the 8X metal plane for the address and data links. Model three and four augment the baseline interconnect with an L-network to accelerate cache access. Model three implements the early look-up proposal (Section 3.1) and model four implements the aggressive look-up proposal (Section 3.2). Model five simulates the hybrid network (Section 3.3) that employs a combination of bus and point-to-point network for address communication. As discussed in Section 3, the bandwidths of the links in all these simulated models are adjusted such that the net metal area is constant. All of the above optimizations help speed up the address network and do not attempt to improve the data network. To get an idea of the best performance possible with such optimizations to the address network, we simulate an optimistic model (model six) 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.

Model seven employs a network composed of only L-wires and both address and data transfers happen on the L-network. Due to the equal metal area restriction, model seven offers lower total bandwidth than the other models and each message is correspondingly broken into more flits. Model eight simulates the case where the address network is entirely composed of L-wires and the data network is entirely composed of B-wires. This is similar to model four, except that instead of performing a partial tag match, this model sends the complete address in multiple

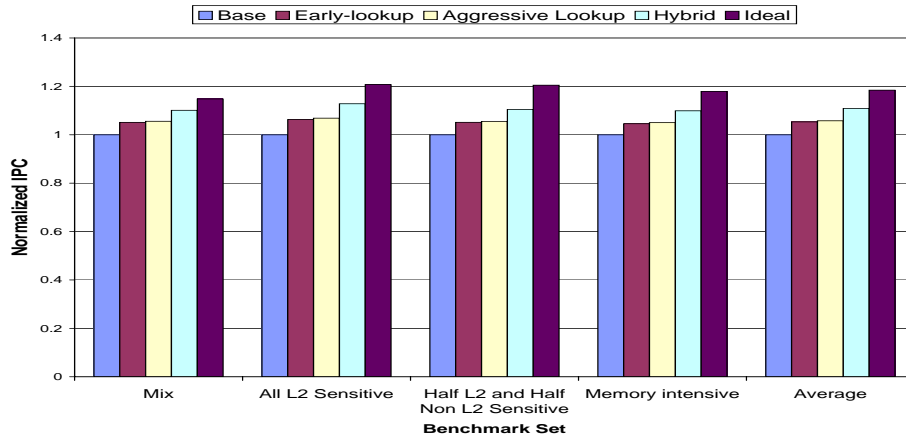
flits on the L-network and performs a full tag match.

Figure 8 shows the IPCs (average across the SPEC2k suite) for all eight processor models, normalized against model one. It also shows the average across programs in SPEC2k that are sensitive to L2 cache latency (based on the data in Figure 1). Figure 9 shows the IPCs for models one through six for each individual program (L2 sensitive programs are highlighted in the figure). Table 5 quantifies the average L2 access times with the proposed optimizations as a function of bank count. In spite of having the least possible bank access latency (3 cycles as against 17 cycles for other models), model one has the poorest performance due to high network overheads associated with each L2 access. Model two, the performance-optimal cache organization derived from CACTI-L2, performs significantly better, compared to model one. On an average, model two's performance is 70% better across all the benchmarks and 112% better for benchmarks that are sensitive to L2 latency. This performance improvement is accompanied by reduced power and area from using fewer routers (see Figure 5(b)).

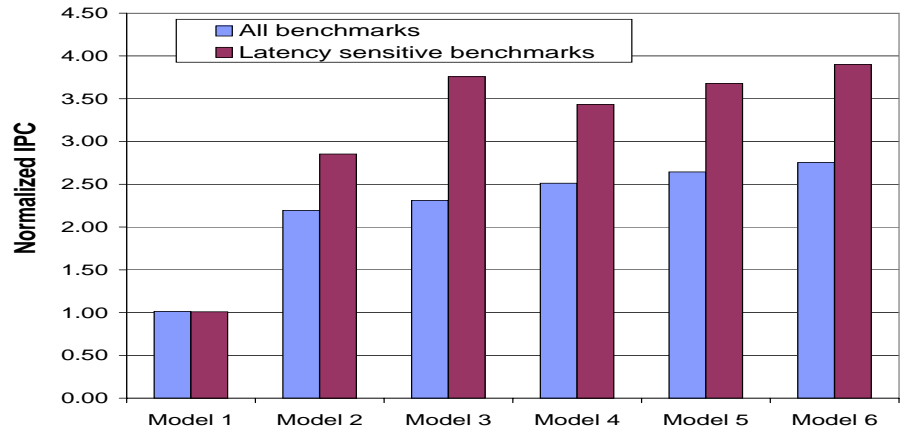
The early look-up optimization discussed in Section 3.1 improves upon the performance of model two. On an average, model three's performance is 6% better, compared to model two across all the benchmarks and 8% better for L2-sensitive benchmarks. Model four further improves the access time of the cache by performing the early look-up and aggressively sending all the blocks that exhibit partial tag matches. This mechanism has 7% higher performance, compared to model two across all the benchmarks, and 9% for L2-sensitive benchmarks. The low performance improvement of model four 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%.

The hybrid model overcomes the shortcomings of model four by reducing the number of routers in the network. Aggressive look-up implemented in a hybrid topology (model five) performs the best and is within a few percent of the optimistic model six. Compared to model two, the hybrid model performs 14% better across all benchmarks and 18% better for L2-sensitive benchmarks.

Model seven employs the L-network for transferring both address and data messages. The performance of this model can be better than the optimistic model (model six) that uses B-wires for data transfers. But the limited bandwidth of the links in model seven increases contention in the network and limits the performance improvement to only a few programs that have very low network traffic. The performance of model seven is comparable to model two (only 2% better, on average). Model eight employs the L-network for sending the complete address in a pipelined fashion. It performs comparably to model four that implements aggressive look-up (4.7% better than model two). But it incurs significantly higher contention on the L-network, making it an unattractive choice for



**Figure 10.** IPC improvement of different cache configurations in an eight core CMP. Benchmark compositions: “Mix” - ammp, applu, lucas, bzip2, crafty, mgrid, equake, gcc; “All sensitive” - ammp, apsi, art, bzip2, crafty, eon, equake, gcc; “Half L2 and Half Non-L2 sensitive” - ammp, applu, lucas, bzip2, crafty, mgrid, mesa, gcc; “Memory intensive” - applu, fma3d, art, swim, lucas, equake, gap, vpr



**Figure 11.** IPC improvement for uniprocessor with 4X wires

CMPs.

Figure 10 shows the IPC improvement of different models in a CMP environment. We evaluate all our models for four different sets of multi-programmed workloads. Set 1 is a mixture of benchmarks with different characteristics. Set 2 consists of benchmarks that are sensitive to L2 hit time. Half the programs in set 3 are L2-sensitive and the other half are not. Set 4 consists of benchmarks that are memory intensive. The individual programs in each set are listed in Figure 10. For our results, we assume that the network bandwidth is doubled to support the increased demands from eight cores. Similar to our uniprocessor results, model one incurs severe performance penalty due to very high network overhead. Model two, derived from CACTI-L2, out-performs model one by 51%. Models three, four, and five yield performance improvements of 5.3% (early look-up), 5.8% (aggressive look-up), and 10.9% (hybrid network), respectively over model two. If the network bandwidth is the same as in our uniprocessor simulations, these performance improvements are 3.0%, 3.28%, and 6.15%, respectively.

As a sensitivity analysis, we report the overall IPC improvements for models two through six for the uniprocessor model with 4X-B wires instead of 8X-B wires in Figure 11. Since 4X wires are slower, the effect of optimizations are more pronounced than in the 8X model. This is likely the expected trend in future technologies where wires are slower, relative to logic delays.

## 5. Related Work

A number of recent proposals have dealt with the implementation of large NUCA caches [6, 11, 12, 17, 18] and focus on optimizing logical policies associated with a baseline cache design. For example, many of these papers employ some form of dynamic-NUCA (D-NUCA), where blocks are allowed to migrate to reduce communication distances. D-NUCA policies also apply to cache models that incorporate the interconnect optimizations proposed in this paper. To the best of our knowledge, only three other bodies of work have attempted to exploit novel interconnects at the microarchitecture level to accelerate cache access. Beckmann and Wood [5] employ transmission lines to speed up access to large caches. Unlike regular RC-based wires, transmission lines do not need repeaters and hence can be directly routed on top of other structures. This property is exploited to implement transmission line links between each cache bank and the central cache controller. The number of banks is limited to the number of links that can be directly connected to the controller. Low-latency *fat* RC-based wires have been employed to speed up coherence signals in a CMP environment [10] and L1 cache access in a clustered architecture [2].

A recent paper by Li et al. [20] proposes the implementation of a NUCA cache in three dimensions. A three-dimensional grid topology is employed and given the low latency for inter-die communication, a dynamic time division multiplexing bus is employed for signal broadcast across dies.

Kumar et al. [19] examine interconnect design issues associated with chip multiprocessors. They detail the scalability problem associated with a shared bus fabric and explore the potential of a hierarchical bus structure.

## 6. Conclusions and Future Work

Delays within wires and routers are major components of L2 cache access time. By considering network parameters in a preliminary tool (CACTI-L2), we derive cache organizations that differ significantly (in terms of performance and power) from those assumed in prior work. Having thus derived an optimal baseline cache organization, we propose novel optimizations to the address network and bank access pipeline that help hide network delays. These optimizations leverage heterogeneity within the network and improve upon the IPC of the baseline by 14% across the SPEC benchmark suite.

This paper has focused on the design of a NUCA cache shared by all cores on a chip. Private L2 cache organizations are also being considered by industry and academia – each core is associated with a large L2 cache and a

request not found in a local cache may be serviced by a remote cache [4, 8, 12, 24, 32, 36]. A remote L2 hit now has a non-uniform access time depending on the remote cache where the block is found and the network delays incurred in communication with the directory and the remote cache. The interconnect continues to play a major role in such cache organizations and many of the interconnect design considerations for a shared NUCA will also apply to private L2 caches.

Our discussions so far have only exploited low-latency L-wires to improve performance. As described in Section 2, wires can also be designed to minimize power (while trading off latency). As future work, we will consider techniques to leverage power-efficient wires to reduce interconnect power. For example, prefetch in a CMP-NUCA cache can improve the performance of commercial workloads by 4-17% [6]. If the prefetch is timely enough, the prefetch operation can happen on a data network that is composed entirely of low power, high bandwidth, high latency wires. Similarly, writeback messages from the L2 cache, or block swaps in a dynamic-NUCA are non-critical operations and can be effected on power-efficient wires. Thus, exploiting wires at microarchitectural level has good potential and can be beneficial in terms of both power and delay.

## References

- [1] H. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.
- [2] R. Balasubramonian, N. Muralimanohar, K. Ramani, and V. Venkatachalapathy. Microarchitectural Wire Management for Performance and Power in Partitioned Architectures. In *Proceedings of HPCA-11*, February 2005.
- [3] K. Banerjee and A. Mehrotra. A Power-optimal Repeater Insertion Methodology for Global Interconnects in Nanometer Designs. *IEEE Transactions on Electron Devices*, 49(11):2001–2007, November 2002.
- [4] B. Beckmann, M. Marty, and D. Wood. ASR: Adaptive Selective Replication for CMP Caches. In *Proceedings of MICRO-39*, December 2006.
- [5] B. Beckmann and D. Wood. TLC: Transmission Line Caches. In *Proceedings of MICRO-36*, December 2003.
- [6] B. Beckmann and D. Wood. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proceedings of MICRO-37*, December 2004.
- [7] D. Burger and T. Austin. The SimpleScalar Toolset, Version 2.0. Technical Report TR-97-1342, University of Wisconsin-Madison, June 1997.
- [8] J. Chang and G. Sohi. Co-Operative Caching for Chip Multiprocessors. In *Proceedings of ISCA-33*, June 2006.
- [9] R. Chang, N. Talwalkar, C. Yue, and S. Wong. Near Speed-of-Light Signaling Over On-Chip Electrical Interconnects. *IEEE Journal of Solid-State Circuits*, 38(5):834–838, May 2003.
- [10] L. Cheng, N. Muralimanohar, K. Ramani, R. Balasubramonian, and J. Carter. Interconnect-Aware Coherence Protocols for Chip Multiprocessors. In *Proceedings of ISCA-33*, June 2006.
- [11] Z. Chishti, M. Powell, and T. Vijaykumar. Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures. In *Proceedings of MICRO-36*, December 2003.
- [12] Z. Chishti, M. Powell, and T. Vijaykumar. Optimizing Replication, Communication, and Capacity Allocation in CMPs. In *Proceedings of ISCA-32*, June 2005.
- [13] W. Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2), March 1992.
- [14] W. Dally and J. Poulton. *Digital System Engineering*. Cambridge University Press, Cambridge, UK, 1998.

- [15] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 1st edition, 2003.
- [16] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. *Proceedings of the IEEE*, Vol.89, No.4, April 2001.
- [17] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. Keckler. A NUCA Substrate for Flexible CMP Cache Sharing. In *Proceedings of ICS-19*, June 2005.
- [18] C. Kim, D. Burger, and S. Keckler. An Adaptive, Non-Uniform Cache Structure for Wire-Dominated On-Chip Caches. In *Proceedings of ASPLOS-X*, October 2002.
- [19] R. Kumar, V. Zyuban, and D. Tullsen. Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads, and Scaling. In *Proceedings of the 32nd ISCA*, June 2005.
- [20] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, N. Vijaykrishnan, and M. Kandemir. Design and Management of 3D Chip Multiprocessors Using Network-in-Memory. In *Proceedings of ISCA-33*, June 2006.
- [21] G. Loi, B. Agrawal, N. Srivastava, S. Lin, T. Sherwood, and K. Banerjee. A Thermally-Aware Performance Analysis of Vertically Integrated (3-D) Processor-Memory Hierarchy. In *Proceedings of DAC-43*, June 2006.
- [22] N. Magen, A. Kolodny, U. Weiser, and N. Shamir. Interconnect Power Dissipation in a Microprocessor. In *Proceedings of System Level Interconnect Prediction*, February 2004.
- [23] D. Matzke. Will Physical Scalability Sabotage Performance Gains? *IEEE Computer*, 30(9):37–39, September 1997.
- [24] C. McNairy and R. Bhatia. Montecito: A Dual-Core, Dual-Thread Itanium Processor. *IEEE Micro*, 25(2), March/April 2005.
- [25] S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The Alpha 21364 Network Architecture. In *IEEE Micro*, volume 22, 2002.
- [26] R. Mullins, A. West, and S. Moore. Low-Latency Virtual-Channel Routers for On-Chip Networks. In *Proceedings of ISCA-31*, May 2004.
- [27] L.-S. Peh and W. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proceedings of HPCA-7*, 2001.
- [28] L.-S. Peh and W. Dally. A Delay Model for Router Micro-architectures. *IEEE Micro*, 21(1):26–34, January/February 2001.
- [29] Semiconductor Industry Association. International Technology Roadmap for Semiconductors 2005. <http://www.itrs.net/Links/2005ITRS/Home2005.htm>.
- [30] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder. Automatically Characterizing Large Scale Program Behavior. In *Proceedings of ASPLOS-X*, October 2002.
- [31] P. Shivakumar and N. P. Jouppi. CACTI 3.0: An Integrated Cache Timing, Power, and Area Model. Technical Report TN-2001/2, Compaq Western Research Laboratory, August 2001.
- [32] E. Speight, H. Shafi, L. Zhang, and R. Rajamony. Adaptive Mechanisms and Policies for Managing Cache Hierarchies in Chip Multiprocessors. In *Proceedings of ISCA-32*, June 2005.
- [33] D. Tarjan, S. Thoziyoor, and N. Jouppi. CACTI 4.0. Technical Report HPL-2006-86, HP Laboratories, 2006.
- [34] H. S. Wang, L. S. Peh, and S. Malik. A Power Model for Routers: Modeling Alpha 21364 and InfiniBand Routers. In *IEEE Micro*, Vol 24, No 1, January 2003.
- [35] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proceedings of MICRO-35*, November 2002.
- [36] M. Zhang and K. Asanovic. Victim Replication: Maximizing Capacity while Hiding Wire Delay in Tiled Chip Multiprocessors. In *Proceedings of ISCA-32*, June 2005.