The Effect of Interconnect Design on the Performance of Large L2 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 have changed the traditional design approach of the memory hierarchy. Many recent proposals advocate splitting the cache into a large number of banks and employ an on-chip network to allow fast access to nearby banks (referred to as Non-Uniform Cache Architectures (NUCA)). While these proposals focus on optimizing logical policies (placement, searching, and movement) associated with a cache design, initial design choices do not include the complexity of the network. With wire delay being the major performance limiting factor in modern processors, components designed without including wire parameters and network overhead will be sub-optimal with respect to both delay and power. The primary contributions of this work are: 1. An extension of the current version of CACTI to include network overhead and find the optimal design point for large on-chip caches. 2. An evaluation of novel techniques at the microarchitecture level that exploit special wires in the L2 cache network to improve performance.

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

1. Introduction

The abundant transistor budget provided by Moore's law enables us to increase the size of on-chip caches to multiple mega bytes. The recently released Intel Itanium processor employs 24MB of on-chip L3 cache and these capacities will continue to increase at future process technologies. While large on-chip caches are effective in reducing the cache miss rate, careful choice of design parameters is critical to translate this reduction in miss-rate to performance improvement.

For the past several years, academic researchers have relied on CACTI [15] to find the optimal design point of on-chip caches. CACTI is an analytical tool that takes a set of cache parameters as input and estimates the access time, layout, and power consumption of on-chip caches. While CACTI is still powerful enough to model small uniform-cache-access (UCA) designs, it does not have the capability to model large caches efficiently. The traditional UCA model has a major scalability problem since it limits the access time of a cache to the access time of its slowest sub-bank. For future large L2 or L3 caches, the disparity in access delay between the fastest and slowest sub-banks can be as high as 47 cycles [14]. Hence, having a single uniform access latency to the entire cache will result in a significant slowdown in performance.

Kim *et al* [14] addressed this problem by proposing a Non-Uniform Cache Architecture (NUCA). The idea is to split the cache into a large number of banks and employ an on-chip network for communication. Now the access time of the cache is a function of distance between the bank and cache controller rather than the latency of the slowest sub-bank. For evaluating their proposals, they assumed a grid network and set up the bank count so that each hop has a latency of one cycle. An ad hoc design such as this may be sub-optimal both in terms of delay and power.

The access latency of a cache depends on delays in the decoders, word-lines, bit-lines, and drivers. Decoder and driver delay components of a cache go up with increase in the number of sub-banks. On the other hand, word line or bit line delay components reduce with decrease in sub-bank size in the vertical and horizontal directions, respectively. The current version of CACTI does an exhaustive search across different sub-bank parameters to calculate the optimal design point so that the net access latency of the cache is minimal. In addition to the above four components, large caches in future processors will have an additional overhead of network delay. Due to the growing disparity between wire and transistor delay, this factor will continue to dominate with technology improvements. The choice of wires in the network link, and the flow control mechanism employed in the network, will have a significant impact on cache access time.

The memory wall problem is well known. Though hierarchical memory and out of order execution help alleviate this problem, L2 access time continues to have a significant impact on processor performance. Figure 1 shows the

*This work was supported in part by NSF grant CCF-0430063 and by an NSF CAREER award.

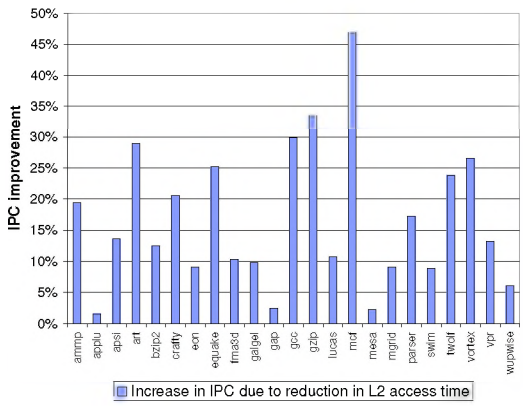

Figure 1. Improvement in IPC due to reduction in L2 access time from 30 cycles to 15 cycles on an aggressive out-of-order processor model.

effect of L2 access time on the SPEC2k benchmark suite. The bar shows the improvement in IPC when L2 access time is reduced from 30 cycles to 15 cycles. The experiment is conducted on an aggressive 8-issue out of order processor model. The details of other processor parameters are discussed in the methodology section. As shown in the figure, even an aggressive eight issue processor suffers significant performance loss with increase in L2 access time. Thus, optimizations targeting a reduction in L2 access time have the potential to improve performance, and hence energy.

In this work, we extend the current version of CACTI to include network parameters and find the optimal design point for large caches. We also show that the choice of wires in a network link has a huge impact on optimal cache parameters and performance. Based on this observation, we propose a heterogeneous network for the L2 cache – a network consisting of special wires with varying latency, power, and bandwidth characteristics. We then present and evaluate novel techniques at the microarchitecture level that exploit these special wires.

The rest of the paper is organized as follows. Section 2 describes the cache area, delay, and power model. Section 3 discusses proposed innovations to exploit wire properties at the microarchitecture level to improve cache access time. These ideas are evaluated in Section 4 and compared against previous work in Section 5. In Section 6, we draw conclusions and discuss other promising avenues for future work.

2. Cache Delay/Power/Area Models

2.1. Wire Model

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 [12]. 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). Our analysis [8] shows that in certain cases, nearly a two-fold reduction in wire latency can be achieved, at the expense of a four-fold reduction in bandwidth. Further, researchers are actively pursuing transmission line implementations that enable extremely low communication latencies [7, 11]. However, transmission lines also entail significant metal area overheads in addition to logic overheads for sending and receiving [4, 7]. 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. Banerjee *et al.* [3] show that at 50nm technology, a five-fold reduction in power can be achieved at the expense of a two-fold increase in latency.

Thus, by varying properties such as wire width/spacing and repeater size/spacing, we can implement wires with different latency, bandwidth, and power properties. Thus, in addition to minimum-width wires (referred to as the baseline *B-Wires*), there are at least three other wire implementations that are potentially beneficial (shown graphically in Figure 2):

- *L-Wires*: Wires that are latency-optimal. These wires employ very wide width and spacing and have low bandwidth.
- *P-Wires*: Wires that are power-optimal. The wires have longer delays as they employ small repeater size and wide repeater spacing.
- *W-Wires*: Wires that are bandwidth-optimal. The wires have minimum width and spacing on lower metal layers and have longer delays.

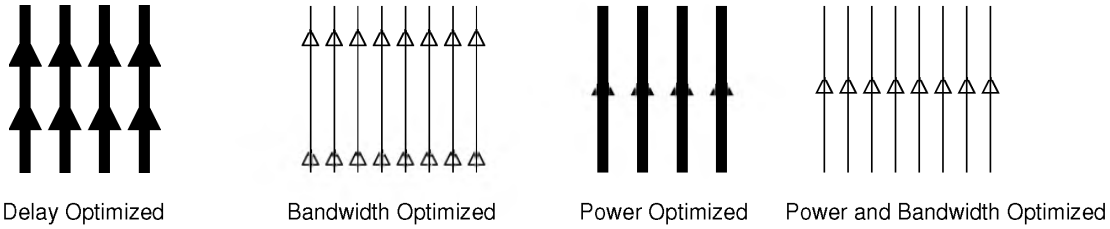

Figure 2. Examples of different wire implementations. Power optimized wires have fewer and smaller repeaters, while bandwidth optimized wires have narrow widths and spacing.

Wire Type	Relative Latency	Relative Area	Wiring Pitch (nm)	Latency (ns/mm)
B-Wire (8X plane)	$1x$	$1x$	210	0.122
B-Wire (4X plane)	$2.0x$	$0.5x$	105	0.244
L-Wire (8X plane)	$0.5x$	$2.5x$	525	0.055

Table 1. Area and delay characteristics of different wire implementations.

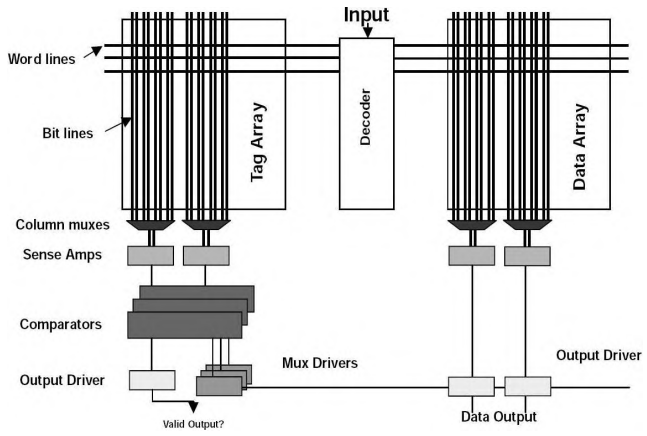

Figure 3. Structure of a cache (from [15]).

For this work we will focus on minimum-width wires in the 4X and the 8X metal planes and the low latency L-wires that have four times the area of an 8X minimum-width wire. Table 1 summarizes the latency, bandwidth and area characteristics of these wires. All the calculations are based on ITRS 2005 roadmap parameters.

2.2. The CACTI-L2 Extension

Figure 3 shows the basic structure of a cache. CACTI divides the total access time of the cache into seven main components (decoder, wordline, bitline, senseamp, comparator, multiplexer-driver, and output driver). Of these, senseamp and comparator delay is almost constant across different cache sizes and its contribution to the total access time reduces with increase in cache size. The mux-driver delay component consists of delay due to multiplexer logic (to select the appropriate line) and driver delay to route the control signal to the output driver. The latter part is proportional to the size of the cache. The decoder part of the cache consists of a single centralized pre-decoder and a separate decoder for each subarray. The output from the pre-decoder is fed to the final decoder stage to drive the appropriate wordline. Thus, the decoder delay component is the sum of time to route address bits to the central pre-decoder, time to route the output of the pre-decoder to the final stage decoder, and the logic delay associated with pre-decoder, decoder, and driver circuits. Thus, decoder delay depends on both cache size and subarray count. The wordline delay of data/tag array is proportional to the length of the array and the bit line delay is proportional to the height of the array. These two delay values can be tweaked by adjusting the aspect ratio of the sub-array. To bring down the delay values of both these components the cache is split into a number of sub-arrays. But, with an increased number of sub-arrays, the latency to send signals to and from the central pre-decoder increases. Thus, there exists a trade-off between the sub-array size and the wire length. CACTI does an exhaustive search across different sub-array counts and aspect ratio values to find the optimal design point.

A similar trade-off exists in large caches between network delay and bank access delay. Figure 4 shows the various delay components of a 32MB cache for different bank count values. We assume a grid network for inter-bank communication and all transfers happen on global 8X wires. Figure 5 shows the L2 organization. The banks are organized such that the number of rows is either equal to number of columns or half the number of columns. The vertical and horizontal hop latencies are calculated using the wire model discussed in the previous sub-section. The bank access value shown in the graph is obtained by feeding the bank size to the unmodified version of CACTI along with other cache parameters. It can be observed that

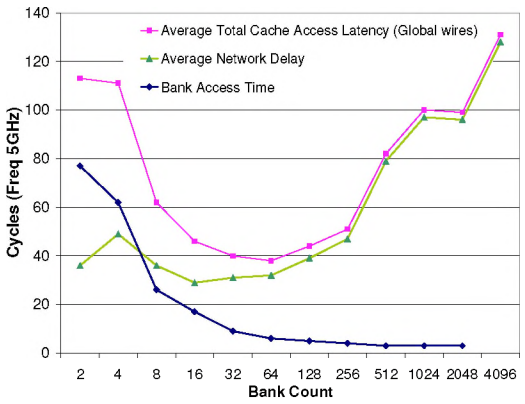

Figure 4. Access latency for a 32MB cache as a function of the number of banks. The cache access latency is the sum of bank access time and network delay.

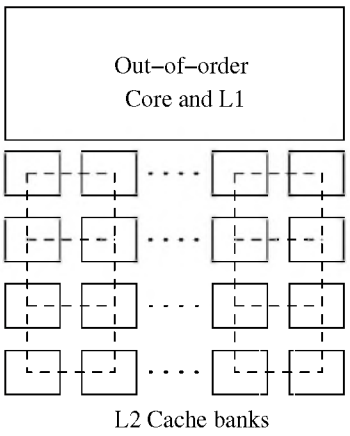

Figure 5. Bank organization.

the bank access latency increases exponentially with a decrease in bank count value. This is because as the size of the cache increases, the decoder delay component (that includes wiring delay) increases significantly. Also, the bank access time saturates beyond a bank count value of 512 (bank size of 64KB). Beyond this point, the latency is primarily due to logic delay associated with each stage, which is constant across different cache sizes.

The average (uncontended) network latency plotted in the graph is obtained by calculating the access time to each indvidual bank and averaging them against the total bank count value. This value depends on both the bank size and total number of banks. It can be observed that the average latency first goes down with an increase in bank count and then starts increasing for large bank count values. If the bank size is extremely large, the hop latency dominates the total access time and hence the network latency is very high. Ideally, the network latency should go down with an increase in bank count. But, dividing the bank into half only reduces the area of data and tag arrays. Other constant area overheads associated with each bank will remain the same and hence the reduction in hop latency is less than half its original value. For very large bank count values, the reduction in hop latency is usually less than the increase in hop count to reach a destination, leading to high average network latencies. The only exception is a change in bank count value from 1024 to 2048 – because hop latencies are rounded up to the next integer value, a doubling in bank count results in halving the vertical hop latency. Thus, finding the optimal bank count value is critical to achieving the least possible access latency.

We extend the current version of CACTI's exhaustive search to include the network delay component for calculating the optimal cache configuration. The third line in the graph shows the total average access time including both bank access latency and network latency. From the graph, it can be inferred that for a 32MB cache, the optimum value of bank count is 32 or 64.

3. Proposed Mechanisms

The previous section details modifications to CACTI and outlines different wire implementations possible in a network. In this section, we discuss microarchitectural techniques that exploit the different wire types introduced in Section 2.

3.1. Early Look-Up

We first discuss techniques to exploit L-wires in the network to improve performance of the cache. The existing baseline network (composed of minimum-width 8X wires) is augmented with a network composed of low-latency L-wires. Due to the high area overhead associated with an L-wire, we assume that each L-network link has a bandwidth of only 24 bits. 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.

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. To break this sequential access, we can send a partial address on the low bandwidth L-network and initiate the cache access. Due to the low latency characteristics of L-wires,

the early look-up starts well before the remaining address reaches the bank. When the remaining address bits reach the bank, they are used for tag comparison before selecting a single cache line and returning it to the cache controller. The transfer of remaining address bits and initial look-up of the cache bank now happen in parallel, resulting in a net reduction in access latency.

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, around 60-70% of the time has elapsed by the time the candidate sets are read out of the appropriate cache bank. By overlapping the two messages 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 number of address bits required to index into the appropriate cache set of a cache bank is typically very small. For the example in Figure 4, for an 8-way set-associative, 512KB cache bank with a block size of 64B, 10 bits are required to select the appropriate set from the bank. Another 6 bits may be part of the message packet to identify the bank. The MSHR id[1] takes up 3 bits and the remaining bits of the 24-bit L-network are used for ECC and control signals.

3.2. Aggressive Look-Up

While the previous proposal is effective in hiding a major part of the cache access time, it still has the drawback of having a huge network delay component associated with each L2 access. We can further improve performance by doing an aggressive look-up instead of just an early look-up. In the above proposal, the L-network carries just enough partial address bits to the destination cache bank to conduct an early look-up. This requires 10 bits of address. In the aggressive look-up technique, we send an additional 8 bits of address on the L-network. Now, after the look-up, instead of waiting for the remaining address to arrive on a slower network, we can carry out a partial tag match on the additional address bits sent on the L-network and aggressively send all matched blocks (along with their tags) to the cache controller. The controller then does the complete tag match to find the appropriate block. This approach reduces the network delay component significantly and hence the net access time at the cost of a slight increase in network traffic and increased complexity at the cache controller. In order to accommodate the 8 additional bits of address for partial tag comparison, the L-network may have to be widened, or the width/spacing of wires may have to be slightly reduced, or ECC signals may have to be omitted. For the evaluation in this paper, we incur an additional metal area cost to send the 8 additional bits.

[1] The Miss Status Holding Register (MSHR) keeps track of outstanding L1 misses.

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, 6 cycles, 4-way word-interleaved
L2 cache	32MB 8-way SNUCA
I and D TLB	128 entries, 8KB page size
Memory latency	300 cycles for the first chunk

Table 2. Simplescalar simulator parameters.

4. Results

4.1. Methodology

Our simulator is based on Simplescalar-3.0 [6] for the Alpha AXP ISA. Table 2 summarizes the configuration of the simulated system. All our delay and power calculations are for a 65nm process technology and a clock frequency of 5 GHz. Contention for memory hierarchy resources (ports and buffers) is modeled in detail. We assume a 32MB on-chip level-2 cache and employ a grid network for communication between different L2 banks. For this preliminary analysis, we evaluate all our proposals on a uniprocessor system.

We simulate six different processor models with different cache configurations. The first model is based on methodologies in prior work [14], 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 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 only employs minimum-width wires on the 8X metal plane for the interconnect (for both, address and data). Model three and four augment the baseline interconnect with a 24-bit wide L-network to accelerate cache access. Model three

Model	Hop latency (v,h)	Bank access time	Bank count	Network link contents	Description
Model 1	1,1	3	512	B-wires	Based on prior work
Model 2	2,2	6	64	B-wires	Derived from CACTIL2
Model 3	2,2	6	64	B-wires & L-wires	Implements early lookup
Model 4	2,2	6	64	B-wires & L-wires	Implements aggressive lookup
Model 5	2,2	6	64	B-wires & L-wires	Implements optimistic case
Model 6	1,1	6	64	L-wires	Latency-bandwidth tradeoff

Table 3. Summary of different models simulated. The bank count and hop latencies are for global 8X wires.

implements the early look-up proposal (Section 3.1) and model four implements the aggressive look-up proposal (Section 3.2). Model five simulates an optimistic case in which the request carrying the address magically reaches the appropriate bank in one cycle. Thus, the L2 access latency of this model primarily depends on the network delay of the return message. This acts as an upper bound for the performance of all other models. Model six employs a network made of only L-wires and both address and data transfers happen on the L-network. The bandwidth of each link in the network is modeled such that the total metal area of the link is the same for both model six and model three (that implements early lookup). Due to this restriction, model six offers lower total bandwidth than the other models and each message is correspondingly broken into more flits. Table 3 summarizes all the simulated models and their cache configurations.

4.2. IPC Analysis

Figure 6 shows the IPCs of different processor models for the SPEC2000 benchmark suite. Workloads that are senstive to cache access latencies are highlighted. In spite of having the least possible bank access latency (3 cycles as against 6 cycles for other models), Model 1 has the poorest performance due to high network overheads associated with each L2 access. Model 2, whose cache parameters are derived from CACTIL2, performs significantly better, compared to Model 1. On an average, Model 2's performance is 10.9% better across all the benchmarks and 16.3% better for benchmarks that are sensitive to L2 latency. This performance improvement also comes with reduced power and area complexity due to fewer routers.

The early-look-up optimization discussed in Section 3.1 improves upon the performance of Model 2. On an average, Model 3's performance is 14.4% better, compared to Model 1 across all the benchmarks and 21.6% better for L2 access time sensitive benchmarks. Model 3's performance improvement comes at the cost of additional metal area overhead due to the L-network. The comparison of models that consume equal metal area is left for future work.

Model 4 further improves the access time of the cache by doing the early look-up and aggressively sending all the blocks selected by the partial address sent on the L-network. This mechanism has 17.6% higher performance, compared to Model 1 across all the benchmarks, and 26.6% for the L2 senstive benchmarks. In addition to Model 3's complexity, this scheme incurs additional traffic in the network because of extra blocks sent due to partial tag matches. The increase in network traffic for all the benchmarks due to false positive matches is less than 1%.

Model 6 employs the L-network for transferring both address and data messages. The performance of this model can be better than the optimistic model (model 5) that uses only B-wires for data transfers. But the limited bandwidth of the links in model six increases contention in the network and limits the performance improvement to only a few programs that have very low network traffic. The performance improvement in model 6 compared to model 1 is 16.2%.

To show the effect of wire parameters on cache design, we simulated all the above models, with a network consisting of global minimum-width wires on the 4X metal plane (instead of the 8X metal plane). The L-networks in models 3 and 4 remain the same as before. Figure 7 shows the average IPC values for different models, normalized against Model 1. Since 4X wires are slower, the effect of optimizations to the L2 are more pronounced. This can be the expected trend in future technologies where wires are slower, relative to logic delays. With the aggressive look-up policy, the average IPC improvement across the benchmark set is about 40% and very close to the upper bound represented by Model 5. Tables 4 and 5 summarize the average cache access latencies (in cycles) for all models as well as the bank latency for wires in the 8X and 4X metal planes, respectively. These values are shown as a function of the number of banks.

5. Related Work

A number of recent proposals have dealt with the implementation of large caches [5, 9, 10, 13, 14]. Most of these papers focus on optimizing logical policies associated with a cache design. 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 et al. [4] employed transmission lines to speed up access to large caches. Unlike regular wires, transmission lines do not need repeaters and hence can be directly routed on top of other structures. Beckmann et

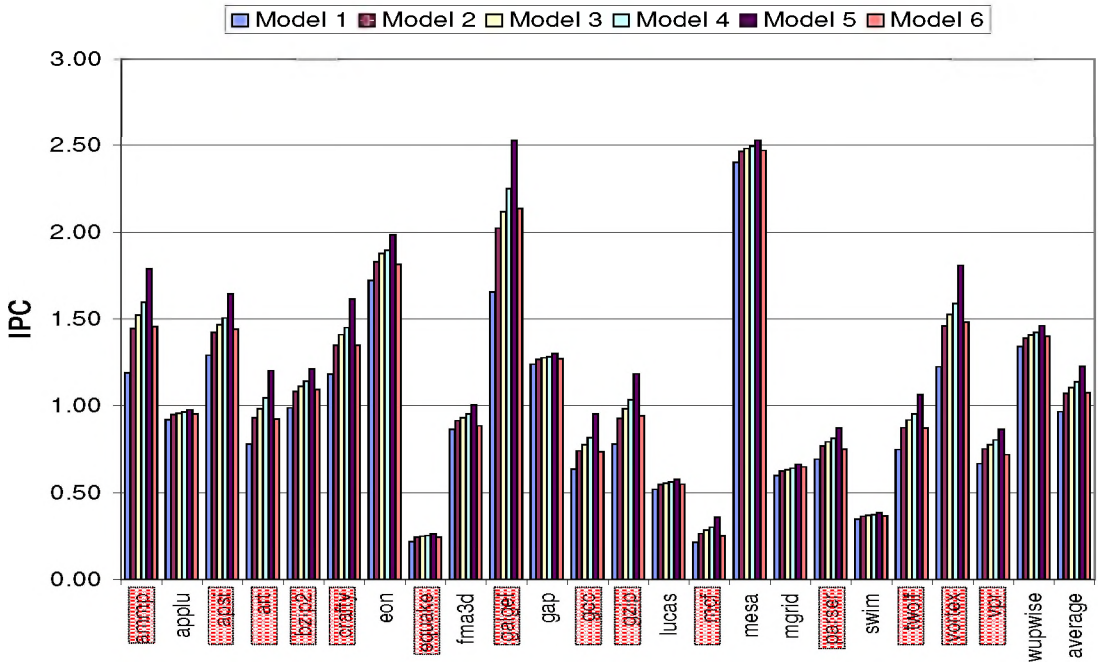

Figure 6. IPCs of SPEC2000 benchmarks for different L2 cache configurations (B-wires implemented on the 8X metal plane).

Bank Count	Bank Access Time	Average Cache Access Time	Early Lookup	Aggressive Fetch	Optimistic Model
2	77	113	96.5	96.5	96
4	62	111	88.5	88.5	87.5
8	26	62	47.6	47.0	45.0
16	17	46	36.5	35.5	32.5
32	9	40	34.0	30.5	25.5
64	6	38	33.3	30.0	23.0
128	5	44	40.6	36.5	25.5
256	4	51	48.4	43.5	28.5
512	3	82	79.2	66.5	43.5
1024	3	100	97.1	83.5	52.5
2048	3	99	99.0	99.0	52.0
4096	3	131	131.0	131.0	68.0

Table 4. Access latencies for different cache configurations. The message transfers are assumed to happen in 8x wires.

Bank Count	Bank Access Time	Average Cache Access Time	Early Lookup	Aggressive Fetch	Optimistic Model
2	77	147	113.5	113.5	113
4	62	156	118.5	111	110
8	26	92	73.6	62	60
16	17	75	63.3	50	47
32	9	71	64.2	46	41
64	6	63	58.0	42.5	35.5
128	5	68	64.1	48.5	37.5
256	4	83	80.0	59.5	44.5
512	3	113	110.1	82	59
1024	3	133	130.0	100	69
2048	3	162	159.1	130.5	83.5
4096	3	196	193.0	163.5	100.5

Table 5. Access latencies for different cache configurations. The message transfers are assumed to happen in 4x wires.

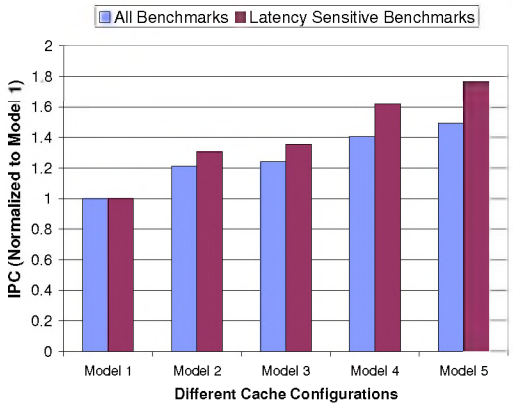

Figure 7. IPCs of SPEC2000 benchmarks for different L2 cache configurations (B-wires implemented on the 4X metal plane).

al. exploited this property and employed transmission line links for connecting each cache bank to the central cache controller. Their choice for number of banks is limited to the number of links that can be directly connected to the controller. In our prior work, heterogeneous wires are employed to speed up L1 cache access in a clustered architecture [2] and to speed up coherence signals in a CMP environment [8].

The recent version of CACTI [15] has support for modeling simple multi-banked caches. But their model is still bus based and assumes a separate bus for each bank. This basic model might be useful for modeling moderately sized caches but their approach is not scalable for future large sized caches.

Kim et al. [14] proposed Dynamic-NUCA to reduce the average access time. In static NUCA, least significant bits of the block address are used to map a cache block to a fixed cache bank. While this approach has the least possible overhead, some of the frequently used blocks might end up getting mapped to banks that are far away from the cache controller and hence could suffer from long hit latencies. DNUCA alleviates this problem by moving the cache block closer to the controller on every hit. This optimization is orthogonal to our proposals and can supplement our proposals by reducing the access time.

Chishti et al. [9] proposed NuRAPID for large on-chip caches. Their multi-banked model is similar to the above mentioned CACTI 3.0 with each bank having dedicated buses for transfering address and data signlas. In NuRAPID, tag arrays carry additional information about the position of the block. Tag and data arrays are accessed sequentially to enable flexible placement of blocks. Their choice of having a separate bus for each bank is not scalable for future large caches.

6. Conclusions and Future Work

The performance of future large L2/L3 caches will be severely constrained by the interconnect. In this paper, we take a first step in extending the popular CACTI tool to model network properties. We show that such a detailed model yields an organization significantly different from that assumed in prior work. We also show that the choice of wire parameters can impact the optimal organization. We propose and evaluate two techniques to accelerate cache access that take advantage of a lower-bandwidth lower-latency network. The first technique initiates early indexing into the cache bank, while the second technique uses partial tag comparison to aggressively forward cache blocks back to the controller. These techniques can improve performance by up to 20%, but incur a cost in terms of metal area.

This paper has only focused on techniques to improve performance. As discussed in Section 2, wires can also be designed to provide low power and high bandwidth (while incurring a performance penalty). As future work, we will also explore optimizations to reduce power consumption. For example, data prefetches and writebacks can happen on power-efficient wires. In a dynamic-NUCA organization [14], block swapping can happen on power-efficient wires, while block search can happen with partial bits on low-latency L-wires. We also observe that the latency benefit of an L-wire is not completely exploited due to the switching circuits in the L-network. An L-wire is capable of transferring signals up to 8mm in a cycle, but each hop on the network is much shorter. We plan to investigate the use of a hybrid L-network that incorporates a bus in the point-to-point interconnect.

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 and D. Wood. TLC: Transmission Line Caches. In *Proceedings of MICRO-36*, December 2003.

[5] B. Beckmann and D. Wood. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proceedings of MICRO-37*, December 2004.

[6] D. Burger and T. Austin. The Simplescalar Toolset, Version 2.0. Technical Report TR-97-1342, University of Wisconsin-Madison, June 1997.

[7] 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.

[8] 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.

[9] Z. Chishti, M. Powell, and T. Vijaykumar. Distance Associativity for High-Performance Energy-Effi cient Non-Uniform Cache Architectures. In *Proceedings of MICRO-36*, December 2003.

[10] Z. Chishti, M. Powell, and T. Vijaykumar. Optimizing Replication, Communication, and Capacity Allocation in CMPs. In *Proceedings of ISCA-32*, June 2005.

[11] W. Dally and J. Poulton. *Digital System Engineering*. Cambridge University Press, Cambridge, UK, 1998.

[12] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. *Proceedings of the IEEE*, Vol.89, No.4, April 2001.

[13] 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.

[14] 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.

[15] 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.