A Domain-Specific On-Chip Network Design for Large Scale Cache Systems

Yuho Jin Eun Jung Kim
Department of Computer Science
Texas A&M University
{yuho,ejkim}@cs.tamu.edu

Ki Hwan Yum
Department of Computer Science
University of Texas, San Antonio
yum@cs.utsa.edu

Abstract

As circuit integration technology advances, the design of efficient interconnects has become critical. On-chip networks have been adopted to overcome scalability and the poor resource sharing problems of shared buses or dedicated wires. However, using a general on-chip network for a specific domain may cause underutilization of the network resources and huge network delays because the interconnects are not optimized for the domain. Addressing these two issues is challenging because in-depth knowledges of interconnects and the specific domain are required.

Recently proposed Non-Uniform Cache Architectures (NUCAs) use wormhole-routed 2D mesh networks to improve the performance of on-chip L2 caches. We observe that network resources in NUCAs are underutilized and occupy considerable chip area (52% of cache area). Also the network delay is significantly large (63% of cache access time). Motivated by our observations, we investigate how to optimize cache operations and design the network in large scale cache systems. We propose a single-cycle router architecture that can efficiently support multicasting in on-chip caches. Next, we present Fast-LRU replacement, where cache replacement overlaps with data request delivery. Finally we propose a deadlock-free XYX routing algorithm and a new halo network topology to minimize the number of links in the network.

Simulation results show that our networked cache system improves the average IPC by 38% over the mesh network design with Multicast Promotion replacement while using only 23% of the interconnection area. Specifically, Multicast Fast-LRU replacement improves the average IPC by 20% compared with Multicast Promotion replacement. A halo topology design additionally improves the average IPC by 18% over a mesh topology.

1. Introduction

With the current rate of technology advancement, increasing wire delays in modern microprocessor designs [1, 12] lead to various technologies to minimize the impact of slow on-chip communication. Typically, on-chip communication has been conducted via shared buses or dedicated wires. Dedicated wire networks can provide the best customization to applications. However, predicting the delays until the late-stage in the design cycle is difficult [18]. These interconnects are influenced by various parasitic capacitances and crosstalks from adjacent wires, which cannot be predicted until the actual layout and routing are performed. For shared bus networks, the whole bus is occupied by a single communication even if multiple communications could operate simultaneously on different parts of the bus. Using global buses is not an effective scalable solution because the bus bandwidth may become a major bottleneck as the number of components on the chip increases.

Another way to design an on-chip communication is with a switched network. When using a switched on-chip interconnection network, all the components are connected to the network that routes packets among them, which has the advantages of structure, performance, and modularity [7, 4, 29, 9]. There has been much research on the architectures of future chip multiprocessor (CMP) designs [28, 24, 20] using switched networks for better scalability and resource sharing. Furthermore, these networks have been adopted to overcome wire delay in specific domains such as large scale cache designs [17, 3].

The regular topologies, such as meshes and tori, have been used in on-chip network designs. However, a general-purpose network with regularly distributed network resources can cause problems in the following two cases; underprovision and overprovision of network resources. Underprovision of network resources causes poor performance. On the other hand, when network resources are overprovided, underutilization of the network resources occurs and large network delays are caused by the increased network size. Furthermore, the overprovision of network resources results in the waste of the chip area. Therefore, it is critical to design an optimal network for a specific domain by breaking the regularity of the interconnection network. It is also important to exploit the potential parallelism of the interconnection networks in the problem domain. Achieving these two goals requires profound knowledge of those areas; interconnects and the specific domain.

In some large scale cache designs [17, 3], 2D mesh networks have been adopted to interconnect small cache banks

This work was supported in part by NSF grants CCF-0541360 and CCF-0541384.

to overcome wire delays. For example, in Non-Uniform Cache Architectures (NUCAs) [17], the cache is broken into multiple banks that can be accessed at different latencies through an on-chip network. D-NUCA (Dynamic NUCA) allows cache blocks to migrate among cache banks in such a way that recently accessed cache blocks can move towards the core, which helps reduce the average cache access time. However, the network delay is still a dominant portion of the cache access time. A 16MB D-NUCA configuration using a 16×16 mesh network demonstrated an average access time of 17 cycles with SPEC2000 benchmarks without network contention while the bank access time is only 3 cycles. The worst case is when the requested cache block is not found in the L2 cache. In this case, all the cache banks in the bank set [1] must be checked sequentially and then the memory is accessed. Although D-NUCA can use partial tag comparisons to detect cache miss early, additional memory in the cache controller is required to store the partial tags. Moreover, 20% of the links in a mesh network are never used, while the network occupies 52% of the total area.

While on-chip networks are relatively well understood for multiprocessor systems [13, 28], there is no prior research considering the detailed design of interconnection networks for such large scale cache designs. Thus, the main purpose of this paper is to investigate the design space of the interconnection framework and, particularly, how it interacts with the rest of the multi-bank cache architecture. The research proceeds as follows: First, we propose a single-cycle wormhole router architecture that supports multicasting efficiently. Because multicasting can significantly reduce the large network delay, its reduces the cache access time dramatically. Unlike the existing multicast routers proposed before, the router takes only one cycle in each hop and does not require any extra storage. It becomes the basic building block of the interconnection network design for the proposed large scale cache systems.

Next, we present Fast-LRU replacement, where cache replacement overlaps with tag-matching in the system. We investigate detailed operations in the network including tag-matching, replacement, and placement for a cache hit and miss. Fast-LRU can be further improved with the multicasting support of the underlying network. The proposed networked cache system shows the best performance when it uses Multicast Fast-LRU replacement.

Finally, we propose a deadlock-free XYX routing algorithm and a new *halo* network topology to reduce the cache access time and minimize the number of links in the cache system. We also discuss the layout of the L2 cache on the processor die to help reduce the cache access time. In the XYX routing algorithm, the normal XY routing is used for delivering cache requests from the core to the banks while the replies from each bank to the core are transfered in the Y direction first. Compared with XY routing, the XYX routing algorithm saves most horizontal links in a mesh network. The removal of horizontal links leads to the removal of all the associated input buffers and the simplification of both the arbiter and the crossbar designs. This simplified router also brings latency reduction as well as reduction in the total chip area. With the halo topology, all the closest banks of each column in a mesh network are placed in the same one-hop distance from the core.

Simulation results with SPEC2000 benchmarks show that the networked cache system with all the proposed techniques improves the IPC by an average of 38% and uses only 38% of the interconnection area of the D-NUCA system with Multicast Promotion replacement. Specifically, the average IPC is improved by 20% with the Multicast Fast-LRU replacement in a mesh topology and by 18% with the halo topology.

This paper is organized as follows: Section 2 describes the related work. Section 3 explains a single-cycle wormhole router architecture with multicasting support and Fast-LRU replacement policy. In Section 4, we develop a new deadlock-free XYX routing algorithm and propose a halo topology for large scale cache systems. Section 5 and Section 6 discuss the simulation platforms and the experimental results, followed by the concluding remarks in Section 7.

2. Related Work

Several works explore the large on-chip cache designs to overcome the wire delay problem. NUCA [17] shows that the traditional large cache based on partitioned subbanking is ineffective because its access latency is determined by the slowest (i.e. farthest from the core) subbank. One of the proposed designs, Static NUCA (S-NUCA), uses dedicated wires to each bank and incurs significant area overhead. In another design (D-NUCA) that uses a switched network, they exploited cache block migration, partial-tag search for early miss detection, and multicast for fast bank access. NuRAPID [5] leverages sequential tag-data access and decouples tag and data placement to save power. It divides the cache data array into several large distance groups and places frequently accessed data to the fast group. However, it has overhead to maintain pointer structures in the existing bank architecture. The communication medium in TLC [2] is a fast transmission line having short inductive-capacitance (LC) delays. Although transmission line can provide express channels, its application is limited due to the high area requirement. Recently, all three designs were examined for cache sharing in a CMP architecture [3, 6, 16].

An on-chip network that enables low-latency for high bandwidth communication is used in partitioned architectures such as CMPs [28, 24] and Network-on-Chips (NoCs) [7, 4, 29, 9]. A few research showed that the network design must be tailored to the communication behavior of applications running on the network to save resources and achieve performance improvement. A mesh network was optimized to each parallel application by analyzing its requirement of both links and routers [13]. Instead of the uniform assignment of buffering resources in a 2D mesh network, the allocation of different buffering size in each channel reduces the implementation costs and increases performance in the

[1] To implement a set associative cache in a networked cache system, a set is distributed across multiple banks. This is named a bank set [17].

NoC design of media applications [15]. Moreover, the router design directly impacts the performance of the network. In a pipelined router architecture, speculative switch allocation reduces the latency by doing virtual channel allocation and switch allocation at the same cycle [23]. Pre-computation of arbitration decisions also reduces the latency by separating the arbitation logic from the critical path [21].

To support "one-to-many" communication primitives, a multicast router has been designed especially for large networks such as multistage networks and 2D mesh/hypercube topologies [27, 19]. Due to the nature of multicasting, we need to provide extra storage to hold packet and a deadlock prevention mechanism. In a chip-to-chip domain, the central-buffer-based design has better performance over the input-buffer-based design, because the queuing capability of the central buffer is superior for unicast packets [27]. However, the additional storage requirement is not desirable in the on-chip domain where area budget is very tight. There are two ways to prevent deadlock; deadlock avoidance/recovery and complete packet buffering. While deadlock avoidance routing causes resource underutilization and deadlock recovery mechanism is highly complex, complete buffering requires the large buffer storage. Therefore, the main challenge of multicast support in an on-chip network is to prevent deadlock without extra storage requirements.

3. System Architecture

In this section we describe a single-cycle multicasting wormhole router that is the basic building block of the interconnection network that connects the core, the banks, and the off-chip memory. Then, we analyze the communication patterns for LRU replacement in a cache network and propose Fast-LRU replacement.

3.1. Single-Cycle Multicasting Router

In this research, we use wormhole routers due to small buffer requirement and high throughput. Figure 1 shows the major components of a wormhole router. It has 5 Physical Channels (PCs) that connect four neighbors and one injection/ejection unit, and each PC is divided into multiple Virtual Channels (VCs). VCs from the same PC share one crossbar switch port to reduce the complexity of the switch. While middle/tail flits require 3 operations (input buffering, switch allocation, and switch traversal), head flits require additional two operations (routing and VC allocation).

Even though an aggressive design can merge adjacent operations into a single-cycle operation, this dependency still exists. Moreover, recent results show that the operating clock cycle for the router is 12 fanout-of-four (FO4) delays [21], which is close to the optimal pipeline delay of the modern superscalar processor [14]. To minimize and break this serial dependency, we use lookahead routing [10], buffer bypassing, speculative switch allocation [23], and arbitration precomputation [21]. All these techniques work well in a lightly loaded network since they can find the possible cases

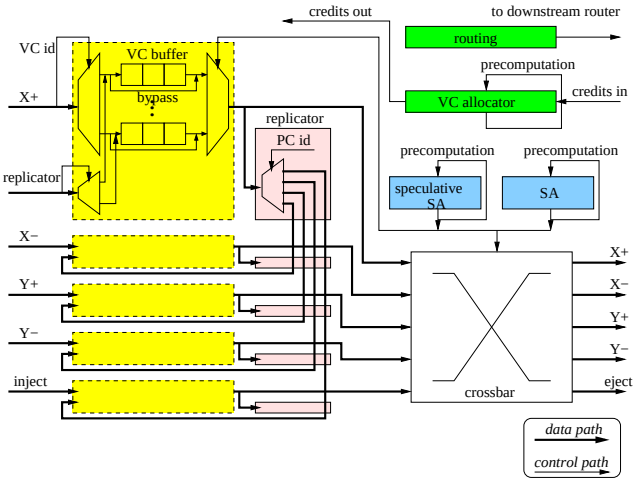


Figure 1. Single-Cycle Multicasting Router

frequently. This router design enables a flit to route in a single clock cycle by reducing the critical path in a traditional pipelined router.

We recognize that multicasting plays a vital role in deciding a cache hit/miss in the networked cache system because it helps access the multiple banks concurrently. Multicasting requires the replication of flits inside the router to forward them to multiple destinations. *Synchronous replication* copies flits after reserving all destination ports in a lock-step, which can result in deadlock. *Asynchronous replication* allows the router to forward flits to a subset of the destination ports. While synchronous replication does not require extra buffers, asynchronous scheme needs additional buffers in order to hold flits until all copies are transmitted. We cannot apply both replication schemes in the router design directly for the following reasons: With the tight chip area constraint, asynchronous replication is not a desirable option; however, synchronous replication in wormhole switching is susceptible to deadlocks. We aim to design a replication scheme without extra buffers and to handle each replica separately in asynchronous manner.

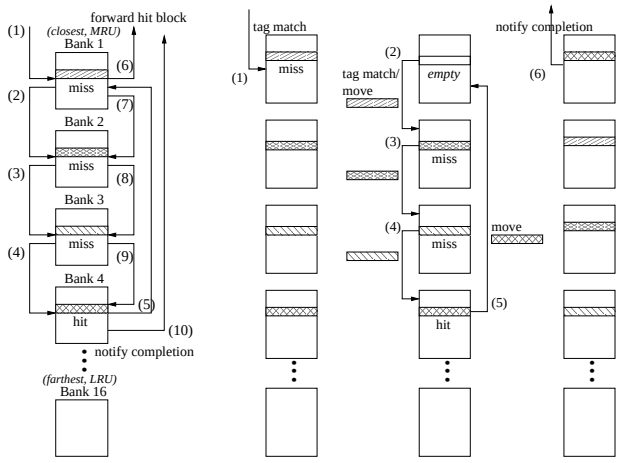
Communication patterns in the networked cache system show that PCs in some routers within the network are not fully utilized. (Detailed analysis is given in Section 4.) This property makes us choose a hybrid scheme that exploits the underutilized input buffer space to store replicated flits. The router shown in Figure 1 copies the original flit to one VC of a different PC, when a multicasting packet needs replication. When a replicator selects a PC that has at least one free VC, PCs that are less frequently used are preferred. Then, one free VC is chosen. A free VC of another PC can be easily obtained by checking the status of the input buffer. If there is no available VC for replication, any flit forwarding is blocked. We observe that blocking rarely happens in the cache systems. This hybrid scheme minimizes the overhead of multicasting support in a low latency router since the changes in the existing VC/switch allocators is not required, and only replication logics are needed to find a free VC of other PCs and connect-

ing wires to their input buffers.

3.2. Fast-LRU Replacement

In this section, we propose *Fast-LRU replacement* with multicasting to reduce the long latency of the LRU replacement scheme after examining optimization of the LRU replacement policy with unicasting.

We start with the brief discussion of cache operations in D-NUCA [17]. As shown in Figure 2 (a), the cache is broken into multiple banks that can be accessed through a mesh network. One column of the mesh represents one set of a set-associative cache, which is statically determined by the low-order bits of the block address. Cache blocks in a set are spread across multiple banks in the column. Each set distributed in a column is called a *bank set*. Thus, the cache system searches for a block by first selecting the column, selecting the set within the column using direct-mapping, and finally performing a tag-match on distributed blocks. Note that each way has a different network latency depending on the distance from the core. Although a bank set can be distributed on every column to give approximately equal access time across all bank sets, we do not consider this configuration due to the larger hop count.



(a) LRU (b) Fast-LRU

Figure 2. Two Unicast LRU Replacement Schemes in a Networked Cache System

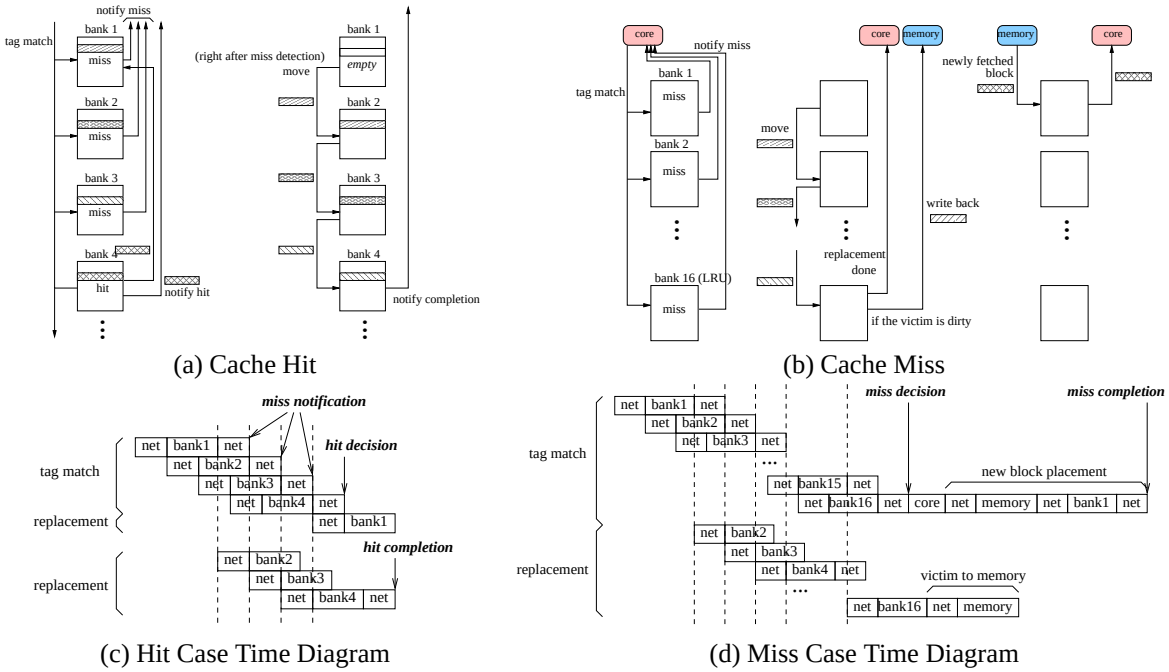
To reduce the average access time, we should place frequently used data in the banks closer to the core, which can be achieved with LRU replacement. The LRU generates 14% higher cache hit rate than Promotion [17], which swaps the hit block with a block in the bank that is next closest to the core. Therefore, it makes the first way and the last way be placed on the closest (MRU) bank and the farthest (LRU) bank, respectively. However, maintaining the LRU order in a bank set requires many swaps of blocks between banks.

Figure 2 (a) shows the overhead of the LRU replacement policy by depicting the required communications among banks. Assuming that a data request is a hit in Bank 4, the request traverses from Bank 1 to Bank 4 ((1) $\sim$ (4)). Then a hit block is sent to Bank 1 ((5)), resulting in the corresponding blocks in Bank 1, 2, and 3 being moved to Bank 2, 3, and 4 ((7) $\sim$ (9)), respectively. In this example, the total communication time is 21 hops including the notification of completion; the initial tag-match time to find a hit bank is 7 hops and the remaining part is 14 hops. It is clear that the cache hit latency can be decomposed into two parts— tag-match and move (replace) operations. Therefore, the total communication time for block movement after finding a hit can easily exceed the initial tag-match time. A cache miss needs tag-match along all banks and a new block placement to the MRU bank incurs multiple block movements to arrange corresponding blocks in the LRU order.

The proposed Fast-LRU replacement allows the tag-match operation to overlap with the replace operation as shown in Figure 2 (b). If there is a miss in a bank, the corresponding block in the bank is evicted and immediately transferred to the next bank with the data request ((1) $\sim$ (4)). Unless the request is a hit in the MRU bank, the corresponding block in the bank is pushed to the next bank consecutively until the final LRU bank. Once there is a hit in a bank, that block is transferred to the MRU bank where its corresponding frame is already empty ((5)). If all the banks generate misses, the request is forwarded to the off-chip memory. Then this new block is stored in the MRU bank and sent to the core. Since the invariant property of LRU is that all the banks ahead of the hit bank generate misses, the total communication time of the Fast-LRU replacement scheme is 12 hops in Figure 2 (b). In addition, this scheme almost halves the number of bank accesses since both tag-match and replacement are performed simultaneously.

Next, we further investigate how to reduce the cache access time by exploiting the multicast router proposed in Section 3.1. Even though Fast-LRU replacement reduces the hit latency, a hit on the LRU bank or a miss on a cache (i.e. all banks produce misses.) still suffer from the long latency, which is the sum of the bank access time over all banks in a bank set and the network latency. Multicasting relieves this problem by allowing concurrent accesses of multiple banks for tag-match.

Figure 3 (a) illustrates a cache hit with Multicast Fast-LRU replacement. Time diagram of each operation is illustrated in Figure 3 (c). When the multicast router attached to the MRU bank (Bank 1) receives a data request, it forwards the request to two destinations, the MRU bank and the second MRU bank (Bank 2), at the same time. The router attached to the second MRU bank (Bank 2) also forwards the request to the attached bank and the next bank (Bank 3), and so on. If the requested block is found in the MRU bank, no block replacement is required and the core is notified of a cache hit. Otherwise, the MRU bank initiates Fast-LRU replacement by sending its evicted block to the second MRU bank. Each non-MRU bank that experienced a cache miss waits for the evicted cache block from the previous bank. As soon as it receives the evicted block, it also sends its evicted block to the next bank. This operation stops at the bank where the requested block is



(a) Cache Hit (b) Cache Miss

(c) Hit Case Time Diagram (d) Miss Case Time Diagram

Figure 3. Fast-LRU Replacement with Multicasting Support

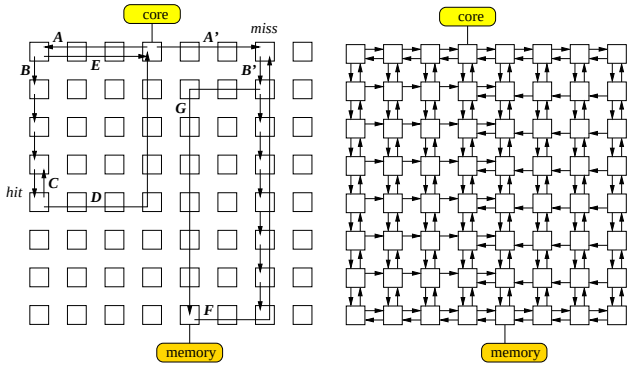
found. Note that the replacement packet (including an evicted block) can never catch the data request packet and that the data request packet always reaches the LRU bank. Each non-MRU bank should not initiate its evicted block transfer to the next bank unless it receives an evicted block from the previous bank.

A cache miss occurs when all the banks in the bank set produce misses as depicted in Figure 3 (b) and (d). It incurs the off-chip memory access and the write-back of the evicted block from the LRU bank (Bank 16) to the memory, if it is dirty. The core waits for all the banks to report misses, and then invokes the memory access. When the memory sends a new block to the MRU bank, the MRU bank sends this newly incoming block to the core. The block movement is similar to that of a cache hit, except the LRU bank notifies the core of the replacement completion and writes back the victim to the memory if the block is dirty.

4. Network Topology and Layout

In this section, we analyze the utilization of the interconnection network for large scale cache systems and explore a few alternatives for simplification and optimization.

Figure 4 (a) shows all possible XY routing communication patterns of a large scale L2 cache system on an 8×8 mesh network. The core is attached to the fourth router of the top row and the memory is attached to the fifth router of the bottom row. Forwarding of a data request to the appropriate bank set needs the traversal of the first row (**A**) and the traversal of banks within the column (**B**) until there is a hit in the bank. When a bank sends the requested cache block to the core, the communication path is **D** or **E** from the hit bank. The data movement between two banks occurs only in a one column of the mesh (**B** or **C**). While a cache miss has the same communication patterns (**A'** and **B'**) as a cache hit, it requires the delivery of the new data block from memory to the MRU bank (**F**). After a new memory block placement to the MRU bank, the evicted block is replaced with the block in the next farther bank (**B'**) or is written back to the memory (**G**), if dirty.



(a) Communication Patterns (b) Minimal Link Requirement

Figure 4. An 8×8 Mesh Network for a Large Scale Cache System

One observation of these patterns is that the unidirectional horizontal link is sufficient except the first row, the last row, and links between the core-attached column and the memory-attached column. Figure 4 (b) shows the minimum set of

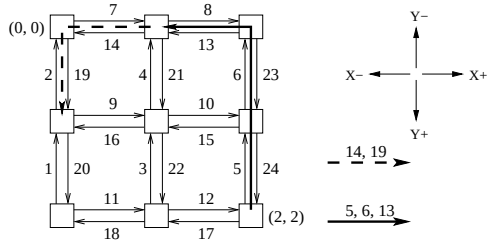
links after removing all unnecessary links. In general, we can remove $(n-2)^2$ links among the total $4(n-1)^2$ links of the $n \times n$ mesh, which makes the link area reduced by 25%.

Another observation is that the horizontal links except the first row are infrequently used, because they are used only when a bank needs to communicate with the core (i.e. the cache controller) or the memory. Considering the locality behavior in the cache, utilization of the horizontal links of the last row is low because those links are only used for the memory accesses. The number of these underutilized links[2] is $n^2 - 2$.

These underutilized links can be eliminated at the expense of small bandwidth loss. This simplified mesh achieves an additional 25% savings in link area. However, it causes the change of the routing scheme since communications from the banks to the core/memory start in the Y direction first, which violates the existing XY routing. Thus, we propose a new routing scheme called XYX routing to overcome this problem. XYX routing is deadlock-free because we can enforce the total order of channels in the mesh network. Figure 5 (a) shows the XYX routing algorithm and Figure 5 (b) shows the channel enumeration in a 3×3 mesh network for XYX routing. Any path in XYX routing follows increasingly numbered channels such as two paths, (14, 19) and (5, 6, 13).

Inputs: coordinates of source bank(X_{src}, Y_{src})
coordinates of destination bank(X_{dest}, Y_{dest})
Outputs: Selected output $Channel$
Procedure:
$X_{offset} := X_{dest} - X_{src}$;
$Y_{offset} := Y_{dest} - Y_{src}$;
if $Y_{offset} \geq 0$ **then**
  if $X_{offset} > 0$ **then** $Channel := X+$;
  else if $X_{offset} < 0$ **then** $Channel := X-$;
  else if $Y_{offset} = 0$ **then** $Channel := Internal$;
  else $Channel := Y+$;
  endif
else
  $Channel := Y-$;
endif

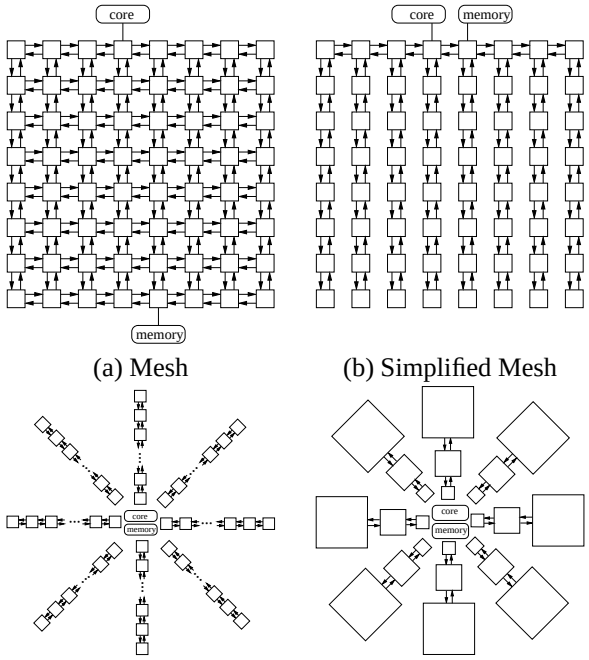
(a) Routing Algorithm



(b) Channel Enumeration in a 3×3 Mesh

Figure 5. XYX Routing

In the view of the router design, expunging horizontal links leads to the elimination of all the input buffers associated with those links and to the simplification of both the arbiter and the crossbar. This simplified router also creates latency reduction as well as saves area.



(a) Mesh (b) Simplified Mesh

(c) Halo Constructed with Uniform Size Banks (d) Halo Constructed with Non-uniform Size Banks

Figure 6. Domain-Specific Development of Network Designs

One of the disadvantages in a mesh network is uneven network latencies among MRU banks depending on the distance from the core. Since there is only one path between the core and one special MRU bank, the leftmost or rightmost MRU banks cannot avoid suffering from the long network latency (Figures 6 (a) and (b)). So it is crucial to provide a direct communication path between the core and each MRU bank. For this purpose, we choose a topology in which the core is located in the same distance from all the MRU banks. We call this topology a *halo* network in Figure 6 (c). The core plays a role of a *hub* to control the departure and the arrival of cache requests. In this design, we assume that the cache controller can support multiple ports/interfaces to the networked cache[3]. A bank set is distributed over multiple banks on a *spike* branched from a hub, which bidirectionally connects all the banks in the order of the way. However, if the size of all banks that build a spike is identical, the banks positioned at the end of the spike cannot fill the increased area entirely. Although we can curve a spike, the spiral spike layout incurs the longer wire delay than the straight spike layout. If the bank size increases along the spike, we can reduce the unused area and draw a compact design by tightly integrating banks on a chip die as shown in Figure 6 (d). As a bank is

[2] In a mesh network requiring the minimal number of links, rows from the second to the last second have n vertical links. The last row has $2(n-1)$ vertical links. Therefore, the total number of these links is $n \times (n-2) + 2(n-1)$.

[3] To issue multiple cache requests simultaneously, we put a small queue (2 entries) for each spike like a multiple issue queue. Thus, a cache request is first stored to each spike queue to be subsequently forwarded to the L2 cache.

located farther from the core, its size becomes larger and its access time increases due to the increased capacity. Therefore, capacity-increased banks have more than one way. A halo network incorporated with non-uniform size banks has a topological benefit by giving the same access time to all the MRU banks, and it enjoys a better area utilization over a halo network with the uniform size banks. Note that the memory controller is in the center of the cache. To access the off-chip memory from the memory controller, its wire delay in the halo is longer than in the mesh.

5. Methodology

We use sim-alpha simulator [8] that models an Alpha 21264 core to generate L2 cache accesses. The clock frequency of the core is scaled to 5 GHz. To measure the contention effect of the banks and the interconnection networks in details, a separate L2 cache simulator with an interconnection network was developed. Sim-alpha directly sends a chunk of L2 accesses to the cache simulator. We models the latency of the bank from Cacti 3.0 [25], and that of the global level wire from the first order RC model [22] under optimal repeater insertion at 65nm technology. We obtain the resistance and capacitance of the unit-length wire from [26], and the wire length is determined by the bank size. Each component of the pipelined router takes one cycle. The base configuration is a 16MB L2 cache by interconnecting 256 64KB banks with a 16×16 mesh network. The core and the memory are attached at the center of the top row and the bottom row, respectively, to evenly distribute traffic. Main parameters are summarized in Table 1.

Table 1. System Parameters

Memory	
Block size	64B
Memory latency (pipelined)	130 cycles + 4 cycles per 8B

Router	
Flit buffer size	4 flits
Number of VCs per PC	4
Flit size	128 bits
Latency of one stage	1 cycle

Wire Delay			
64KB	128KB	256KB	512KB
1 cycle	2 cycles	2 cycles	3 cycles

Bank Access Latency		
bank size	tag matching only	tag matching+replacement
64KB	2 cycles	3 cycles
128KB	4 cycles	4 cycles
256KB	4 cycles	5 cycles
512KB	5 cycles	6 cycles

To measure various design impacts, we use SPEC2000 benchmarks. We fastforward 2 billion instructions, warm up the L2 cache for the next 100 million instructions, and measure the performance of each architecture for remaining instructions. Table 2 shows the perfect L2 IPC and L2 cache access behavior of each benchmark.

A 32-bit address is divided into 4 fields: tag (12 bits), index (10 bits), bank-column (4 bits), and offset (6 bits). The *bank-column* is used to select one of 16 columns of the network while the *index* identifies one of the entries in each bank in the column. With uniform size 64KB banks, each bank is a direct-mapped cache, and the cache blocks in the banks form a 16-way bank set.

Table 2. Benchmarks Used for Experiments

benchmark name	instr. exec.	perfect L2 IPC	L2 read	L2 write	L2 access per instr.
applu(FP)	500M	0.43	9.444M	4.428M	0.028
apsi(FP)	1B	0.40	12.375M	8.204M	0.021
art(FP)	500M	0.40	63.877M	13.578M	0.155
galgel(FP)	2B	0.43	19.415M	4.137M	0.012
lucas(FP)	1B	0.44	19.506M	13.226M	0.033
mesa(FP)	2B	0.40	2.907M	2.656M	0.003
bzip2(INT)	2B	0.39	16.301M	4.233M	0.010
gcc(INT)	500M	0.29	26.201M	14.827M	0.082
mcf(INT)	250M	0.34	29.500M	15.755M	0.181
parser(INT)	2B	0.38	18.257M	6.915M	0.013
twolf(INT)	1B	0.38	20.283M	7.653M	0.028
vpr(INT)	1B	0.41	12.459M	5.024M	0.017

Since a cache network delivers packetized data, the on-chip network does not need the separate address bus and data bus used in the traditional cache. In wormhole switching, flitization requires the overhead data [7] in a flit such as type (2 bits for specifying head/middle/tail), size (7 bits for flit size), routing (8 bits for source and destination), and communication type (1 bit for unicasting/multicasting). Because the width of link is 16B, a read request packet or a notification packet that consists of only the address fits in one flit even with the overhead data. When a packet includes a block data for write request, replacement, memory access, or hit data forwarding, one packet consists of 32-bit address, 64B data and overhead data, which are divided into five flits.

6. Experimental Evaluation

We perform experiments to inspect the efficiency of communication support of Fast-LRU replacement in Section 6.1. Different interconnects for the networked cache system are examined in Section 6.2. The area of router, wire, and cache in each design is analyzed in Section 6.3.

6.1. Performance of Fast-LRU Replacement

Figure 7 shows how the total average cache access latency is divided into bank access, network traversal, and memory access for benchmarks in 16MB L2 cache with uniform size banks. A significant portion of the total average latency is network access (65% on the average) while bank access (25%) and memory access (10%) are relatively small. LRU is a better policy than Promotion since more requests are hit in the MRU (fastest) banks. Specifically, LRU shows a hit increase by 5%-19% at the MRU banks.

In Figure 8, we compare performance results from the Multicast Fast-LRU method with those of two existing Promotion schemes [17] [4] and Unicast LRU/Fast-LRU methods.

[4] In our implementation of cache miss of Promotion, the incoming block

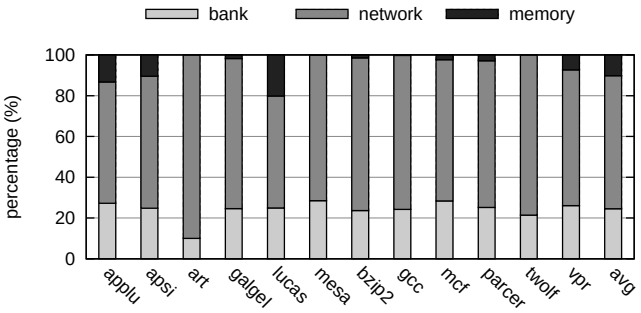


Figure 7. Latency Distributions of L2 Cache Access in the Unicast LRU Environment

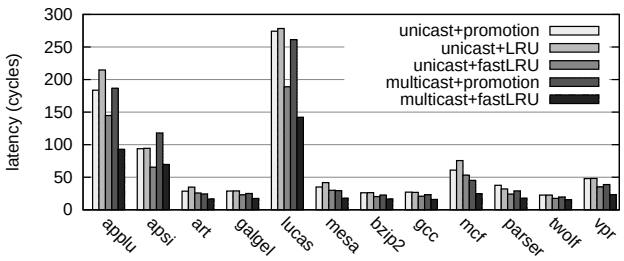
Figure 8 (a) illustrates the average access latency while hit and miss latencies are depicted in Figures 8 (b) and (c) [5]. In the unicast environment, LRU naturally increases the average access latency by 4.4% over Promotion, but Fast-LRU reduces it by 30.2%. Recall that the number of bank accesses in Fast-LRU is almost the same as Promotion, but it concentrates hits to the MRU(closest) banks. All the results of Multicast Fast-LRU show reduction of the average access latency by 46% over Unicast LRU and 27% over Unicast Fast-LRU. Also Multicast Fast-LRU reduces the average hit and miss latency of Unicast LRU by 48% and 32%, respectively. Its latency improvement over Multicast Promotion is 37%, and the IPC is improved by 20%.

6.2. Performance Comparison of Different Network Designs

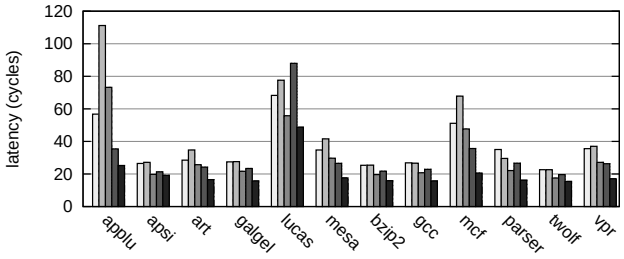
We examine the performance of the L2 cache with varying network size, network topology, bank size, and the position of the core and the memory. We evaluate six designs summarized in Table 3. All configurations have the same capacity (16MB) and use efficient Multicast Fast-LRU replacement. Design A, the baseline configuration, uses a 16×16 mesh network to connect 256 64KB banks. Design B uses the same size simplified network by removing most horizontal links and moving the memory controller next to the core shown in Figure 6 (b). A small 16×4 mesh network with large banks is incorporated in Design C. Design D still uses a mesh network, but non-uniform size banks are used. Non-uniformity of the size still maintains the same associativity but has different wire delays between tiles. One bank set is constructed by 5 banks: two 1-way 64KB banks, one 2-way 128KB bank, one 4-way 256KB bank, and one 8-way 512KB bank in the order of the distance from the core. In 16×5 mesh, we set the same delay (3 cycles) in the horizontal direction as for 512KB bank while the delay in the vertical direction increases as the bank size increases. Designs E and F use the halo topology such as Figures 6 (c) and (d). Since the memory controller is located in the center of the cache system, we need to consider the longer wire delay for the off-chip memory access. Wire delays are 16 and 9 cycles in Designs E and F, respectively.

from the memory evicts the data in the closest bank and causes recursive replacement. In [17], if the incoming block replaces the data in the cache and the victim is moved to the memory (zero-copy) or the lower-priority bank (one-copy), the miss latency can be reduced. However, it can evict the important data from the cache.

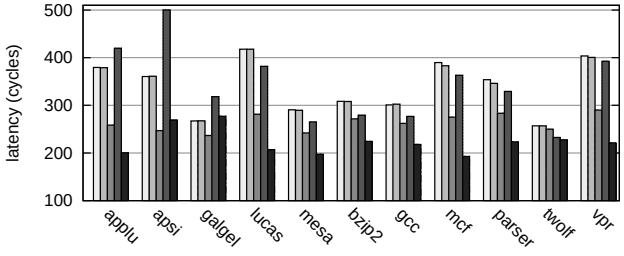
[5] We omit *art* results in Figure 8 (c) since there is no cache miss except compulsory misses during our simulation.



(a) Average Access Latency



(b) Average Hit Latency



(c) Average Miss Latency

Figure 8. L2 Cache Access Latency Comparisons

Table 3. Different Network Designs

Design	Interconnection Network	Bank Size
A	16×16 mesh	uniform (64KB)
B	16×16 simplified mesh	uniform (64KB)
C	16×4 simplified mesh	uniform (256KB)
D	16×5 simplified mesh	non-uniform
E	16-spike halo (length of spike=16)	uniform (64KB)
F	16-spike halo (length of spike=5)	non-uniform

Figure 9 shows the relative IPC normalized to the Design A. The simplified mesh network in Design B achieves almost the same performance results as Design A despite the decreased bandwidth. Even low hit rate benchmarks, *applu* and *lucas*, show the IPC increase by almost 7% and 10%.

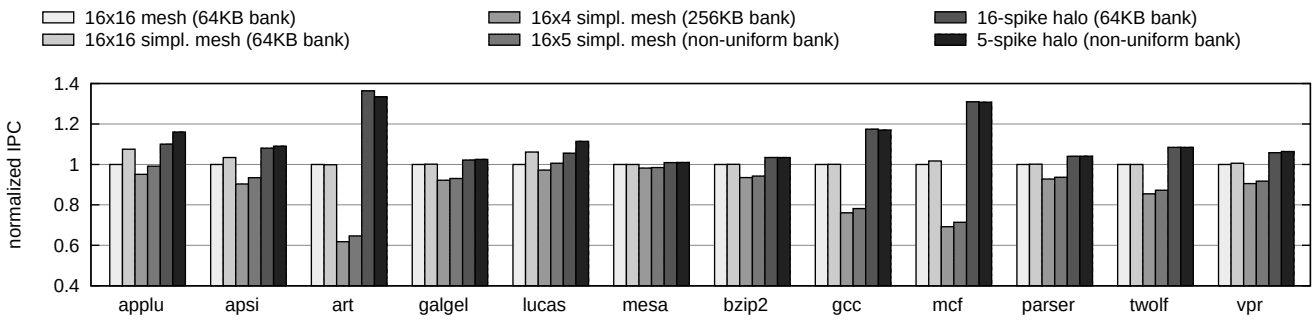


Figure 9. Performance Comparison in Different Interconnection Networks

The main reason of the performance enhancement is the miss latency reduction. Designs C and D show the average performance degradation by 14% and 12% respectively, due to the high wire latency to traverse the larger bank. The halo topology in Designs E and F gives performance improvement by 12% and 13%. Since non-uniform size banks can reduce the wire delay to the memory, Design F shows slightly better performance than Design E, especially in *applu*, *apsi*, and *lucas*. However, *art* having no misses in our simulation shows performance degradation due to increased wire delay for large size banks. Design F achieves 1.13 times the IPC increase over Design A. We can observe this improvement in both high and low hit rate benchmarks (1.33 times in *art* and 1.19 times in *lucas*). Compared with NUCA's Multicast Promotion, the halo topology coupled with Multicast Fast-LRU improves 1.38 times of the IPC.

6.3. Area Comparison of Different Network Designs

We estimate the required area on the banks, routers, and links of a 16MB L2 cache system. The bank area is extracted from Cacti model [25]. In the router area, we account for flit buffers and the crossbar switch, where each is analytically driven by the feature size [11]. We estimate the link area by computing its width and length. Assuming the wire pitch is $1\mu m$, a bidirectional link that transmits 128-bit flit consists of 256 wires, which has $256\mu m$ width. To estimate the link length to cover one tile, we use the sum of both router and bank areas. We assume that there is no additional area for repeaters and latches in a wire because wires are not routed over banks.

Table 4. Area Analysis of Network Designs

Design	bank (%)	router (%)	link (%)	L2 area (mm^2)	chip area (mm^2)
A	47.8	20.8	31.4	567.70	567.70
B	58.4	13.0	28.6	464.60	521.99
E	67.5	14.1	18.4	402.30	1602.22
F	78.7	5.7	15.7	312.19	517.61

Table 4 describes the area consumption of each component for four designs discussed in Section 6.2. The last column is the size of the minimal rectangular chip that includes the L2 cache. Design A (16×16 mesh) uses almost 52% of the cache area for the network. Design B (16×16 simplified mesh) consumes the 18% smaller area than Design A by removing almost half of the links and incorporating the simple 3-port router that takes up only 48% of the normal router area. For halo networks (E, F), we assume that a $4mm \times 4mm$ core is placed in the center of a L2 cache. Design E (a halo network connecting uniform size banks) reduces the 13% of the L2 cache area over Design B, but its L2 cache uses only about a quarter of the total die. Applying the non-uniform size banks (Design F) not only reduces 6.3 times the unused area in a die over Design E, but also consumes the only 23% of the L2 area of Design A. The main reason for its compact layout is the small size of network that requires fewer routers and links. Area estimation for Design F is based on the configuration shown in Figure 10.

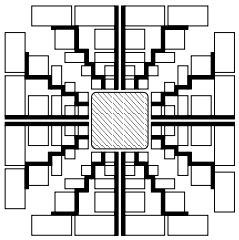


Figure 10. 16-Spike Halo Network Design for L2 Cache

7. Concluding Remarks

We have presented in this paper a domain-specific on-chip network design for large scale L2 cache systems. This research was motivated by the discernment that the network latency is significant, while the network is occupying considerable chip area in networked cache systems. The detailed design proposed in this paper includes: (i) a single-cycle router architecture with multicasting support as the basic building block of the interconnection networks; (ii) Fast-LRU replacement that can reduce the network latency; (iii) appropriate deadlock-free XYX routing algorithm that requires no horizontal links in a mesh except the first row so as to save area and power; (iv) a new network topology, called a *halo* network, where the MRU banks are of the same distance from the core; and (v) a halo network with non-uniform

sized banks, thus reducing the wasted area on the processor die. Simulation results performed on SPEC2000 benchmarks show that the proposed cache system with all the proposed techniques achieves significant performance improvement and area efficiency.

The important conclusions of this work are the following: First, domain-specifically designed networks for large cache systems can provide better performance than general interconnection networks once we understand the communication patterns well. Next, the average cache latency of the networked cache systems is comprised of three factors: cache bank access latency, network latency, and memory latency. Among them, the network latency occupies the largest portion; therefore, reducing the average cache latency requires a reduction in the network latency. Finally, domain-specific designs cannot only improve the performance, but also can reduce the area claimed by the cache systems.

We are planning to expand the study presented in this paper to include CMP environments by first analyzing the traffic patterns and finding suitable interconnects for those systems. Another direction for future work is energy consumption analysis of the networked cache systems. We are developing an on-demand power control scheme that can dynamically turn on/off a subset of cache systems. The proposed multicast router cannot be directly adopted in a network for general workloads, because we cannot guarantee the availability of input buffers for replication. The multicast router design in a general on-chip network should be revisited.

References

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures. In *Proceedings of ISCA*, pages 248–259, 2000.

[2] B. M. Beckmann and D. A. Wood. TLC: Transmission Line Caches. In *Proceedings of MICRO*, pages 43–54, 2003.

[3] B. M. Beckmann and D. A. Wood. Managing Wire Delay in Large Chip-Multiprocessor Caches. In *Proceedings of MICRO*, pages 319–330, 2004.

[4] L. Benini and G. D. Micheli. Networks on Chips: A New SoC Paradigm. *IEEE Computer*, 35(1):70–78, 2002.

[5] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Distance Associativity for High-Performance Energy-Efficient Non-Uniform Cache Architectures. In *Proceedings of MICRO*, pages 55–66, 2003.

[6] Z. Chishti, M. D. Powell, and T. N. Vijaykumar. Optimizing Replication, Communication, and Capacity Allocation in CMPs. In *Proceedings of ISCA*, pages 357–368, 2005.

[7] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proceedings of DAC*, pages 684–689, 2001.

[8] R. Desikan, D. Burger, S. Keckler, and T. Austin. Sim-alpha: a Validated, Execution-Driven Alpha 21264 Simulator. Technical Report TR-01-23, The University of Texas at Austin, Department of Computer Sciences, 2001.

[9] M. Forsell. A Scalable High-Performance Computing Solution for Networks on Chips. *IEEE Micro*, 22(5):46 –55, Sep/Oct 2002.

[10] M. Galles. Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip. In *Proceedings of Hot Interconnect*, 1996.

[11] B. T. Gold. Balancing Performance, Area, and Power in an On-Chip Network. Master's thesis, Department of Electrical and Computer Engineering, Virginia Tech, August 2004.

[12] R. Ho, K. Mai, and M. Horowitz. The Future of Wires. In *Proceedings of the IEEE*, pages 490–504, 2001.

[13] W. H. Ho and T. M. Pinkston. A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns. In *Proceedings of HPCA*, pages 377–, 2003.

[14] M. S. Hrishikesh, D. Burger, S. W. Keckler, P. Shivakumar, N. P. Jouppi, and K. I. Farkas. The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays. In *Proceedings of ISCA*, pages 14–24, 2002.

[15] J. Hu and R. Marculescu. Application Specific Buffer Space Allocation for Networks–on–Chip Router Design. In *Proceedings of ICCAD*, November 2004.

[16] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler. A NUCA Substrate for Flexible CMP Cache Sharing. In *Proceedings of ICS*, pages 31–40, 2005.

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

[18] M. Lajolo, M. S. Reorda, and M. Violante. Early Evaluation of Bus Interconnects Dependability for System-on-Chip Designs. In *Proceedings of Fourteenth International Conference on VLSI Design*, pages 371–376, 2001.

[19] X. Lin and L. M. Ni. Deadlock-Free Multicast Wormhole Routing in Multicomputer Networks. In *Proceedings of ISCA*, pages 116–125, 1991.

[20] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz. Smart Memories: a Modular Reconfigurable Architecture. In *Proceedings of ISCA*, pages 161–171, 2000.

[21] R. D. Mullins, A. West, and S. W. Moore. Low-Latency Virtual-Channel Routers for On-Chip Networks. In *Proceedings of ISCA*, pages 188–197, 2004.

[22] R. H. J. M. Otten and R. K. Brayton. Planning for Performance. In *Proceedings of DAC*, pages 122–127, 1998.

[23] L.-S. Peh and W. J. Dally. A Delay Model and Speculative Architecture for Pipelined Routers. In *Proceedings of HPCA*, pages 255–266, 2001.

[24] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore. Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture. In *Proceedings of ISCA*, pages 422–433, 2003.

[25] P. Shivakumar and N. P. Jouppi. Cacti 3.0: An Integrated Cache Timing, Power and Area Model. Technical report, Compaq Computer Corporation, 2001.

[26] SIA. International Technology Roadmap for Semiconductors, 2003. `http://public.itrs.net`.

[27] C. B. Stunkel, R. Sivaram, and D. K. Panda. Implementing Multidestination Worms in Switch-Based Parallel Systems: Architectural Alternatives and their Impact. In *Proceedings of ISCA*, pages 50–61, 1997.

[28] M. B. Taylor, W. Lee, S. P. Amarasinghe, and A. Agarwal. Scalar Operand Networks. *IEEE Trans. Parallel Distrib. Syst.*, 16(2):145–162, 2005.

[29] G. Varatkar and R. Marculescu. Traffic Analysis for On-Chip Networks Design of Multimedia Applications. In *Proceedings of DAC*, pages 795 –800, 2002.