An Analysis of On-Chip Interconnection Networks for Large-Scale Chip Multiprocessors

DANIEL SANCHEZ, GEORGE MICHELOGIANNAKIS,
and CHRISTOS KOZYRAKIS
Stanford University

With the number of cores of chip multiprocessors (CMPs) rapidly growing as technology scales down, connecting the different components of a CMP in a scalable and efficient way becomes increasingly challenging. In this article, we explore the architectural-level implications of interconnection network design for CMPs with up to 128 fine-grain multithreaded cores. We evaluate and compare different network topologies using accurate simulation of the full chip, including the memory hierarchy and interconnect, and using a diverse set of scientific and engineering workloads.

We find that the interconnect has a large impact on performance, as it is responsible for 60% to 75% of the miss latency. Latency, and not bandwidth, is the primary performance constraint, since, even with many threads per core and workloads with high miss rates, networks with enough bandwidth can be efficiently implemented for the system scales we consider. From the topologies we study, the flattened butterfly consistently outperforms the mesh and fat tree on all workloads, leading to performance advantages of up to 22%. We also show that considering interconnect and memory hierarchy together when designing large-scale CMPs is crucial, and neglecting either of the two can lead to incorrect conclusions. Finally, the effect of the interconnect on overall performance becomes more important as the number of cores increases, making interconnection choices especially critical when scaling up.

Categories and Subject Descriptors: B.4.3 [**Hardware**]: Input/Output and Data Communications—*Interconnections*; C.1.2 [**Computer Systems Organization**]: Multiple Data Stream Architectures—*Interconnection architectures*

General Terms: Design, Performance

Additional Key Words and Phrases: Networks-on-chip, chip multiprocessors, hierarchical networks

This work was supported in part by the Stanford Pervasive Parallelism Lab, the Gigascale Systems Research Center (FCRP/GSRC), the National Science Foundation under Grant CCF-0702341, and the National Security Agency under Contract H98230-08-C-0272. Daniel Sanchez is supported by a Fundacion Caja Madrid Fellowship and a Hewlett-Packard Stanford School of Engineering Fellowship. George Michelogiannakis is supported by a Robert Bosch Stanford Graduate Fellowship.

Authors' addresses: Electrical Engineering Department Stanford University, Stanford, CA; email: {sanchezd, mihelog}@cs.stanford.edu; christos@ee.stanford.edu.

DOI 10.1145/1756065.1736069 http://doi.acm.org/10.1145/1756065.1736069

ACM Reference Format:
Sanchez, D., Michelogiannakis, G., and Kozyrakis, C. 2010. An analysis of on-chip interconnection networks for large-scale chip multiprocessors. ACM Trans. Architec. Code Optim. 7, 1, Article 4 (April 2010), 28 pages
DOI = 10.1145/1756065.1736069 http://doi.acm.org/10.1145/1756065.1736069

1. INTRODUCTION

Due to the scaling difficulties of uniprocessor architectures [Agarwal et al. 2000], chip multiprocessors (CMPs) have become the dominant design approach. Thanks to the increasing transistor budget provided by Moore's Law, it is expected that the number of cores per chip will grow accordingly. CMPs with tens of cores are already being manufactured [Bell et al. 2008; Tremblay and Chaudhry 2008], and chips with hundreds of cores will be available in the near future [Intel 2008].

To connect the increasing number of cores in a scalable way, researchers are evaluating packet-switched networks-on-chip (NoCs) [Dally and Towles 2001; De Micheli and Benini 2002; Owens et al. 2007]. The increasing disparity between wire and transistor delay [Ho et al. 2001] and the dependence between interconnect and memory system performance suggest that the relative importance of NoCs will increase in future CMP designs. As a result, there has been significant research in topologies [Bononi et al. 2007; Kim et al. 2008; Tota et al. 2006], router microarchitecture [Kim et al. 2005; Mullins et al. 2004], wiring schemes [Balasubramonian et al. 2005], and power optimizations [Wang et al. 2003]. Nevertheless, there is a great need for further understanding of interconnects for large-scale systems at the architectural level. Previous studies have focused on small-scale CMPs [Kumar et al. 2005], have used synthetic traffic patterns [Bononi et al. 2007; Kim et al. 2005; Mullins et al. 2004] or traces [Tota et al. 2006], or do not model the other components of the memory hierarchy [Kim et al. 2008].

In this article, we explore the performance, area, and power cost trade-offs of NoCs for large-scale CMPs with up to 128 multithreaded cores. Similarly to a previous study for small-scale CMPs with up to 16 cores and bus-based interconnects [Kumar et al. 2005], we investigate the interconnection network together with the memory system. Specifically, we use full-detail execution-driven simulation to model a homogeneous CMP system with simple multithreaded cores and directory-based cache coherence. We use several diverse applications from the PARSEC [Bienia et al. 2008], SPLASH-2 [Woo et al. 1995], and BioParallel [Jaleel et al. 2006] suites for the evaluation. To trim down the huge design space, we make reasonable assumptions on what constitute realistic design points. We study three realistic topologies (mesh, fat tree [Leiserson 1985], and flattened butterfly [Kim et al. 2007]) under different bandwidth assumptions and also compare them against an idealized network. In order to see the impact of the interconnect on the system, we use chip-level metrics (performance, area, and power), instead of traditional interconnect-centric metrics (e.g., bisection bandwidth).

Using this infrastructure, we attempt to shed light into the following questions that can help guide future work on large-scale CMPs and their interconnects: What is the best network topology in terms of performance, area, and power for a large-scale CMP? Does one topology clearly dominate or is there space for hybrid or reconfigurable designs? What are the key constraints that limit network performance, cost-performance ratio, throughput, and latency? What is the relative importance of the interconnect versus other design decisions, such as the sharing scheme in the memory hierarchy? What is the importance of the interconnect for the overall system? What conclusions can we extract to guide future research in interconnects and memory hierarchies for CMPs?

The main conclusions from our study are the following.

—The interconnect has a large impact on overall performance. Compared to an idealized interconnect, practical topologies triplicate miss latency and increase execution times by up to 76%, severely impacting scalability. Furthermore, the relevance of the interconnect increases with the number of cores.

—For CMPs with up to 128 multithreaded cores, we can build networks with enough throughput for all applications well within reasonable area and power budgets. Hence, network latency, and not throughput or congestion, is the main performance constraint for NoCs in such systems.

—The interconnect topology is the parameter, among the ones explored in this study, that has the largest impact on performance. We identify the flattened butterfly as the best topology among the evaluated ones. For a negligible increase in system area and power budgets, the flattened butterfly outperforms the mesh and the fat tree in all our workloads, with speed-ups of up to 22%.

—Increasing the number of threads per core, which makes each core more tolerant to network latency, slightly decreases the differences across topologies. However, interconnect latency is still the limiting factor in memory system performance, and significant differences between topologies remain.

—Changes in the L2 cache organization affect the traffic offered to the interconnect. To draw meaningful conclusions, cache hierarchy and interconnect must be studied together.

The rest of the article is organized as follows. Section 2 reviews NoCs. Section 3 presents the architectural framework for our study. Section 4 describes the experimental methodology. Section 5 analyzes our evaluation results, while Section 6 summarizes the lessons learned. Finally, Section 7 presents related work, and Section 8 concludes the article.

2. ON-CHIP INTERCONNECTION NETWORKS

Small-scale CMPs use interconnect schemes such as buses, rings, and crossbars [Kumar et al. 2005]. While buses are relatively simple, they suffer from scalability issues as all communication is serialized. Moreover, arbitration for the shared medium can impose a significant latency. Crossbars eliminate serialization by providing a separate path from each source to each destination.

Unfortunately, the area and power costs of a crossbar increase quadratically with the number of network endpoints. For rings, the average hop count is proportional to the number of nodes, and all traffic shares the same links, making bandwidth a possible bottleneck. Hence, none of these approaches is appropriate for large-scale CMPs.

2.1 Background on Scalable Interconnects

Packet-switched NoCs have been proposed as a scalable and modular alternative for large scale CMPs [Dally and Towles 2001; De Micheli and Benini 2002; Owens et al. 2007]. NoCs are composed by a topology of routers connected via point-to-point links. Packets are divided into flow-control digits (flits). Flits may be composed of multiple—but usually one—physical digits (phits), the size of which is defined by the network link width. Thus, packets are transferred across the narrower channels over several cycles, incurring a serialization latency. The head flit carries the destination address that routers use to determine the proper output port and virtual channel (VC) [Dally 1990] for the whole packet. Routing can be either deterministic, always following the same path, or adaptive, taking into account the network state, such as congestion.

Routers are the basic building block of scalable interconnects. They use per-VC input port buffers. Head flits at buffer heads go through routing computation and VC allocation. Nonhead flits are assigned the same output port and VC as their head flit. Flits then proceed to switch allocation. Allocators try to find the best match considering all requests and output port states. Winning flits traverse the switching fabric, which delivers them to the proper output ports. Routers are typically pipelined and several speculation or precomputation techniques are used to reduce the critical path or the latency under light load [Kim et al. 2005; Mullins et al. 2004].

NoCs use VCs to enable deadlock avoidance, optimized channel utilization, improved performance, and quality of service [Bjerregaard and Mahadevan 2006; Dally 1990]. Disjoint traffic classes use separate VCs and routing algorithms are designed to avoid cycles within and across VCs [Duato 1993]. Blocked flits from one VC do not affect flits from other VCs, since they use independent buffering resources. Per-VC credits are used to avoid input buffer overflow. A router can forward flits only if it has credits to consume from the downstream router for that VC. Credits represent free slots in the corresponding next-hop buffer.

2.2 Topology

The topology defines how routers are connected with each other and the network endpoints. For a large-scale system, the topology has a major impact on the performance and cost of the network. In this work, we study three practical topologies suggested for large-scale CMPs: the 2D mesh, the fat tree [Leiserson 1985], and 2D flattened butterfly [Kim et al. 2007]. Figure 1 illustrates the position of routers and point-to-point links in each topology. Table I provides an asymptotic comparison of the three topologies for key metrics. T represents the number of sources and destinations in the network. Topology options are

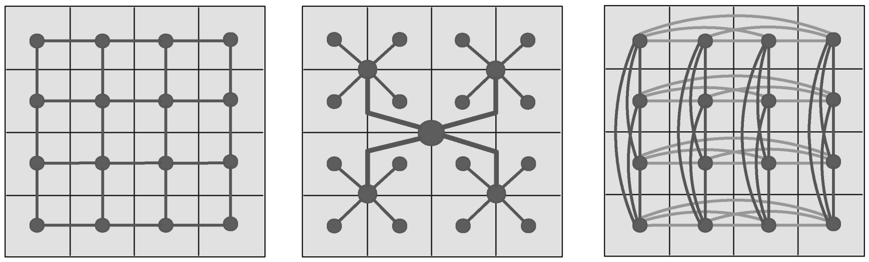


Fig. 1. The three topologies in a 2D layout: 2D mesh (left), fat tree (center), and 2D flattened butterfly (right). The fat tree connects to higher-level nodes using multiple links.

Table I. Qualitative Comparison of the Three Topologies for a CMP System with T Nodes (Cores, Cache Banks, and Controllers)

	Routers (N)	Router I/Os (P)	Bisection BW
2D mesh	$\frac{T}{C}$	$(C+4)$	$\sqrt{N}W$
Fat tree	$\sum_{i=1}^{L} \frac{T}{D^i}$	D^L (root)	$D^L W$
2D FBFly	$\frac{T}{C}$	$(C+2(\sqrt{N}-1))$	$\sqrt{N}W\lfloor\frac{N}{2}\rfloor$

	Hops (worst)	Hops (average)
2D mesh	$2\sqrt{N}$	$\sim\sqrt{N}+1$
Fat tree	$2L$	$\sim\frac{3}{2}L$
2D FBFly	4	3.5

The point-to-point links have a width of W bits. The mesh and flattened butterfly use a concentration factor of C. The fat tree degree is D, hence the tree has $L = \log_D T$ levels. The hop count is the number of link traversals under dimension-order minimal routing. The average number of hops assumes uniformly distributed traffic.

numerous in large-scale systems. Although we have chosen representative, fundamentally different topologies, other design points exist. For example, topologies which have direct links between distant routers, such as express cubes [Grot et al. 2009; Dally 1991] or other low-diameter networks [Xu et al. 2009], can have similar advantages as the flattened butterfly due to their reduced hop count.

2D Mesh. The 2D mesh is a popular interconnect choice in large-scale CMPs [Bell et al. 2008; Intel 2008]. Each of the $\frac{T}{C}$ routers connects to its four neighboring routers and C source or destination nodes. The degree of concentration C, in nodes per router, is typically applied to reduce the number of routers and therefore hops. In this work, we use a mesh with a concentration factor, commonly referred to as a cmesh.

The major advantage of the mesh is its simplicity. All links are short and balanced and the overall layout is very regular. The routers are low radix with up to $C + 4$ input and output ports, which reduces their area footprint, power overhead, and critical path. The major disadvantage is the large number of hops that flits have to potentially go through to reach their final destination (proportional to $\sqrt{N}$ for N routers). Each router imposes a minimum latency (e.g., 3 cycles) and is a potential point of contention. A large number of hops has a direct impact on the energy consumed in the interconnect for buffering, transmission, and control. Hence, meshes could face performance and power scalability issues for large-scale systems. To address this shortcoming, researchers have proposed meshes with physical [Dally 1991] or virtual [Kumar et al. 2008] express links.

Fat Tree. The fat tree connects routers in a tree manner, with sources and destinations at the leaves. A tree with degree D has $log_D T$ levels. To avoid congestion toward the root of the tree, fat trees use an increasing number of point-to-point links per connection. The number of links is multiplied by the tree degree D as we move toward the root. Flits travel upward in the tree until they reach the first common ancestor between the source and the destination. If multiple links are available at a level, one is chosen at random.

The major advantage of the fat tree is the large amount of bandwidth available. The fat links reduce the probability of contention under low load as well. The disadvantage of fat trees is the need for large-radix routers toward the top of the tree. Such routers have higher area and power overheads due to the quadratic increase in the complexity of the internal crossbar. Finally, the fat links contribute to higher leakage power consumption than a mesh.

2D Flattened Butterfly. The 2D flattened butterfly is derived by flattening the routers in each row of a conventional butterfly topology while preserving inter-router connections [Kim et al. 2007]. Routers connect with every other router in each axis. Essentially, this topology provides the connectivity of a mesh with additional links. Thus, in a 4×4 network, each router connects with three other routers in the x axis, and with three others in the y axis. Similarly to the mesh, a concentration factor is typically applied to reduce the router overhead. If the concentration factor in our example is 5 nodes per router, each router is 11×11.

The major advantage of the flattened butterfly is the small number of hops for network traversals under minimal routing. For two dimensions, flits can always reach any node with three or four hops (i.e., two or three routers). Using the longer links minimizes the number of routers visited with their associated latency and energy overheads. The additional links reduce the chance of congestion and provide higher bandwidth as well. The major disadvantage of the flattened butterfly is the high-radix routers, which are expensive in terms of area and power. The larger number of links increases area and leakage power as well.

3. CMP ARCHITECTURE FRAMEWORK

To trim the huge design space for large-scale CMPs, we only consider homogeneous chips with directory-based, cache-coherent memory hierarchies. Such systems scale reasonably well, have been the focus of several academic and industrial efforts, and put significant pressure on the interconnect. Figure 2 presents the CMP organization and Table II summarizes its key parameters. We investigate systems with up to 128 cores, using 64 cores as the default configuration because it can be implemented within reasonable area and power budgets in a 32 nm process (see Section 4). We use fine-grain multithreaded cores to provide tolerance to memory access latency. The L2 cache size was chosen to make its total area roughly equal to that of the cores.

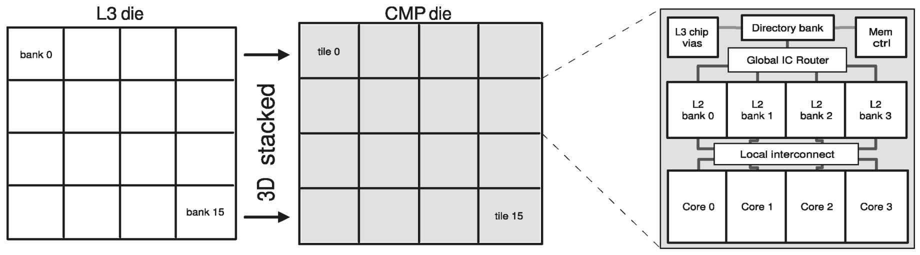


Fig. 2. Tiled CMP design.

Table II. Main Characteristics of the CMP System

Cores	32–128 cores (default **64**), x86 ISA, IPC = 1, 1/**2**/4-way multithreaded
Coherence protocol	Directory-based, MOESI between L1s-L2s and L2s-directory
Consistency model	Sequential consistency
L1 caches	32KB, 4-way set associative, split D/I, 1-cycle latency
L2 cache	256KB - **1**MB per bank, 4 banks/tile, 16-way set associative shared by L1s of the same tile, noninclusive 5-cycle tag/10-cycle data latencies for 1MB banks, pipelined
L3 cache	16MB per bank, 1 bank/tile, 16-way set associative shared across the whole chip, acts as victim cache for L2s 10-cycle tag/21-cycle data latencies, pipelined
Directory	1 bank/tile, idealized
Memory controller	1 controller/tile, single DDR-3 channel

Default values are shown in boldface. The latencies assume a 64-node system implemented on a 32 nm process.

3.1 Base Architecture

We structure the chip in a number of tiles, each with four cores, an L2 cache, an L3 cache bank, a memory directory bank, and a memory controller. The four banks of each L2 cache are shared by the four cores in the tile via the local interconnect. In addition, the L2 and directory banks are directly connected to the tile's global interconnect router. When a request must access an L3 cache bank, a memory directory bank, or a memory controller, the request is routed to the proper tile based on the address interleaving scheme.

The tiled design is motivated by several factors. First, due to temporal and spatial locality, L1 misses are likely to be much more common that L2 misses. Having a fully shared L2 across 64 or more cores would cause an unnecessarily high average L2 access time, even with NUCA designs. Sharing the L2 cache, however, has been shown to be beneficial due to increased hit rates, faster communication between cores that share the L2, and a more balanced cost between maintaining L1 and L2 cache coherence [Huh et al. 2005]. Having a private L2 per four cores is a reasonable compromise. We use a full crossbar as the local interconnect as it is cheap to implement within a tile and allows all possible communication pairs. Bus-based interconnects could face scalability issues for the eight components connected to the local interconnect and may impose significant latency due to arbitration, which is a primary concern for these networks, as we will see in Section 5. Second, tiling provides a concentration factor for the global interconnect of $C = 5$ (the four L2 banks and directory), reducing the number of routers and thus the latency of the global interconnect.

The three-level cache hierarchy uses a full-fledged cache coherence protocol. The L1 caches are write-back, allocate-on-write. The L2 cache is noninclusive with respect to the L1s. For coherence purposes, the L2 maintains a copy of the L1 tags. Coherence between L1s and same-tile L2 and between different L2s is kept using a directory-based MOESI protocol. Data addresses are interleaved across the directory banks of the different tiles, and each bank manages the memory mapped by its own memory controller. To mitigate the high latency of main memory accesses, we include an L3 cache. This cache is shared by the full chip, has a 16MB bank per tile, and acts as a victim cache for the L2s [Jouppi 1990]. We assume that the L3 is implemented on a different die and is stacked on top of the CMP [Benkart et al. 2005], with one set of vias per L3 bank connecting it to the proper CMP tile. To minimize global interconnect traffic, the address interleaving scheme is such that L3 misses are serviced by the memory controller on the same tile.

We model an idealized directory with a fixed 6-cycle latency that keeps a full bit vector of sharers for each cache line. A realistic implementation would use a directory cache per tile with additional directory entries stored in the L3 cache and main memory. Each tile has a memory controller with a single DDR-3 memory channel. For a 64-way CMP, this gives 16 memory channels. This is optimistic, as it would take a high number of pins. A practical alternative would be to model FB-DIMM channels, which can be implemented using a total of 1,120 pins and would offer similar bandwidth at somewhat higher latencies. However, this issue does not affect the accuracy of our results: Due to the large L3 cache used, none of our applications exhausts DRAM bandwidth (as we will see in Section 4.3). For reference, the peak memory bandwidth usage across all the benchmarks in the 64-way CMP is 39GB/s.

3.2 Interconnection Networks

Table III shows the implementation details of the local and global interconnects. We use 3-stage pipelined routers. The first pipeline stage consists of look-ahead routing [Galles 1997] and VC allocation. Look-ahead routing decides the output at the next hop. Thus, head flits enter the router containing their desired output, so that VC allocation can start immediately and in parallel with routing computation. They represent a reasonable design point, since 2-stage routers with a comparable, but still larger, cycle time would require speculative VC allocation, thus complicating the router design [Mullins et al. 2004]. Link latencies depend on the distance between the routers in the global interconnect scheme. The fat tree and flattened butterfly topologies include some short links and some long links. We discuss this issue in Section 4. Flits are composed of

Table III. Main Characteristics of the Interconnection Networks

General	Two separate virtual networks for requests/replies in both global and local networks
Routers	3-stage pipeline: look-ahead routing computation and VC allocation, switch allocation, switch traversal Round-robin separable VC/switch allocators 4 VCs per virtual network Buffering: 144 bytes per VC (8 flits with 18B flits)
Links	9, **18**, or 36B flit size, point-to-point, bi-directional (full-duplex) 1 cycle latency in local interconnect, 2–7 cycle latency in global interconnect

one phit for all network configurations and datapath widths. Their size is equal to the network link width. Since cache lines are 64 bytes, we model 8-byte data requests messages and 72-byte responses. We use an 18-byte (144 bits) flit size by default so that requests are 1 flit and responses 4 flits long.

The fat tree degree is 4. When the system size is such that the root node would only have two children (e.g., in systems with 32 tiles), we connect these children directly and eliminate the root. Also, we use multiple root nodes to avoid routers with a radix larger than 8. For the mesh and the flattened butterfly, we use deterministic dimension-order routing in order to easily prevent cyclic dependencies. Dimension-order routing also helps reduce the latency of routers by simplifying routing computation and thus doing it in the same cycle as VC allocation. Adaptive routing would have minimal benefits for our experiments, as links are not highly utilized.

4. METHODOLOGY

4.1 Simulation

We perform detailed execution-driven simulation of the whole CMP. We use our own simulator to model in-order, fine-grain multithreaded x86 cores with $IPC = 1$ for all operations excluding main memory accesses. Multithreading is used for latency tolerance. Section 5.8 shows why 2-way multithreading is a reasonable default choice for our system. To faithfully model the memory hierarchy and interconnect, we interface to the Wisconsin GEMS toolset [Martin et al. 2005], which includes Princeton's Garnet interconnect simulator [Agarwal et al. 2007]. We only simulate user-level application and library code. Although full-system simulation is possible with Simics and GEMS, it leads to 10 times slower simulation, which is prohibitive for the scale of the CMPs we study. We have modified both GEMS and Garnet to cover our modeling needs.

When comparing the different topologies, we also use an idealized network. This network has a fixed 3-cycle latency in the local interconnect, and a fixed 9-cycle latency in the global interconnect. The idealized network does not have any contention issues and does not require serialization of packets into flits. Hence, it is a tool that helps us measure the overall importance of the interconnect in application performance. We also model an unrealistic

Table IV. Area and Power Projections for Cores and Caches in our 64-core CMP in 32nm

	# per CMP	Component Area	Total Chip Area	Component Power	Total Chip Power
Cores	64	3.20mm^2	205mm^2	0.8W	51.2W
L2 banks	64	2.27mm^2	145mm^2	108mW	6.93W
L3 banks	16	17.20mm^2	275mm^2	153mW	2.44W
Total	–	–	350mm^2(CMP)	–	60.6W
	–	–	275mm^2(L3)	–	

Table V. Area and Power Projections for the Cores based on the Sun Niagara 2 and Intel Atom Designs

	Process	Frequency	Original Core Area	32nm Scaled Core Area
Niagara 2	65nm, 1.2V	1.4GHz	12mm^2	3mm^2
Atom	45nm, 1.1V	1.6GHz	6.8mm^2	3.4mm^2
Estimated	32nm, 0.9V	2.0GHz	–	3.2mm^2

	Chip TDP	Core TDP	32nm Scaled Core TDP
Niagara 2	95 W	5.4 W	1.1 W
Atom	2.5 W	1.1 W	0.5 W
Estimated	–	–	0.8 W

crossbar topology with single-cycle links in the global network to see the relative influence of serialization and contention on performance.

All the simulations are performed with warmed-up caches. Moreover, we use a small random perturbation to main memory latency and do multiple runs per configuration and workload to obtain stable averages [Alameldeen and Wood 2003].

4.2 Area, Power, and Latency Models

For power and area estimations, we use technology parameters based on ITRS predictions for the 32nm technology node [ITRS 2007]. At this process, the 64-core CMP at a frequency of 2GHz has reasonable area and power requirements. Table IV lists the area and power requirements for the major CMP components. We now explain how we estimated these requirements.

Cores. We approximate core area and power by scaling down two existing core designs: the Sun Niagara 2 [Nawathe 2007] and the Intel Atom [Gerosa et al. 2008]. Table V shows their characteristics and scaled-down area and power. We use a 32nm, 0.9V process as predicted by ITRS. For area calculations, we assume that core dimensions scale proportionally to feature size. For power calculations, manufacturers provide only the thermal design power (TDP) of the whole chip, so we approximate the per-core TDP by assuming that the power density in the cores is twice as in the rest of the chip. We compute power as $P = \frac{1}{2}CV_{DD}^2 f$, assume that the switched capacitance scales down with area, and take into account the differences in frequency and voltage. We cannot scale the leakage component of power differently, as manufacturers do not publish its

Table VI. Area, Latency and Power Projections for the L2 and L3 Caches, based on CACTI 5.3

	Area	Tag Latency	Data Latency
1MB L2 bank	2.27mm^2	5 cycles	9 cycles
16MB L3 bank	17.20mm^2	10 cycles	21 cycles

	Energy per read access	Leakage power	Estimated total power
1MB L2 bank	0.37nJ	0.1mW	108mW
16MB L3 bank	1.32nJ	1.4mW	153mW

contribution to overall power. In a 32nm process, a scaled-down Niagara 2 core would measure 3mm^2, while a scaled Atom would measure 3.4mm^2. Therefore, we assume 3.2mm^2 per core to be a reasonable area budget. In terms of power, a Niagara 2 consumes 1.1W versus the 0.5W of the Atom. Differences in power are larger because the Niagara 2 has multithreaded, server-oriented cores, while the Atom is a low-power processor. We can assume that a reasonable power budget for our simple cores is 0.8W per core.

Caches. We estimate the area, latency and power of L2 and L3 caches using CACTI 5.3 [Thoziyoor et al. 2008]. For both caches, we use a single 128-bit read-write port. The L2 cache is implemented in the ITRS-LSTP process (transistors with high threshold voltage to reduce leakage), and the L3 cache die is implemented in a conventional commodity DRAM process. Table VI enumerates the detailed power, area, and latency estimations. We also list power estimations at 2GHz for our applications with highest power consumptions for the L2 (svm) and L3 (canneal). However, note that these estimates may be lower than the worst-case ones, since higher cache utilization can be achieved with especially memory-intensive workloads or a power virus.

Interconnect. To estimate the area and power requirements for the interconnect, we used detailed models from [Balfour and Dally 2006]. We use technology parameters from the 32nm 0.9V ITRS-HP process provided by CACTI 5.3. For the point-to-point links, we use wires in the 4$\times$ plane, which have a 256nm pitch. A reasonable delay/wire length is 110ps/mm. This requires repeaters spaced 514μm and 25$\times$ the size of the smallest inverter. In our channel area models, we include the repeaters and flip-flops, but assume that the 4$\times$ wires can be routed over logic without impacting logic density. This is somewhat optimistic, but obtaining an accurate estimation of the overhead introduced by the wires would require synthesis and layout of the chip. Note that unlike previous work [Kumar et al. 2005], we use narrow point-to-point links instead of wide buses, so it is more likely that a significant percentage of the wire area is hidden. Power models for the links include dynamic and leakage power for repeaters and flip-flops, and clock wire power. The flip-flops in the channels are clock-gated locally. Router area is estimated using detailed floorplans. This includes the crossbar and the buffers, which are modeled as efficient custom SRAMs. Power models also include the control wires throughout the router, clock wire power, and take into account both dynamic and leakage power. However, the power for the allocators is not modeled. This causes only minor inaccuracies, since allocators constitute only a small fraction of router power—for example, 7% as shown in Kahng et al. [2009]. Critical devices in the channels and router datapaths, such as the repeaters driving large wire capacitances, are sized to ensure correct operation at our clock frequency. A more detailed description of these models can be found in Balfour and Dally [2006].

Traditional interconnect-focused studies typically equalize the area, power, or bisection bandwidth of the interconnect to provide a fair comparison between different topologies. However, bisection bandwidth was developed as a fairness metric in off-chip networks, since wires were expensive and required I/O chip pins, but wires are abundant in on-chip networks [Dally and Towles 2001]. Thus, large-scale CMP designs are more likely to be limited by area and power instead of on-chip wiring. Additionally, in the context of a full CMP, only channel widths that cause packets to be an integer multiple of the flit size are reasonable. Therefore, instead of equalizing topology metrics, we choose an acceptable range of flit sizes and evaluate the trade-offs between performance and cost (area and power) in the context of the full chip. Optimizing chip-level metrics is, after all, the overall goal of a CMP design.

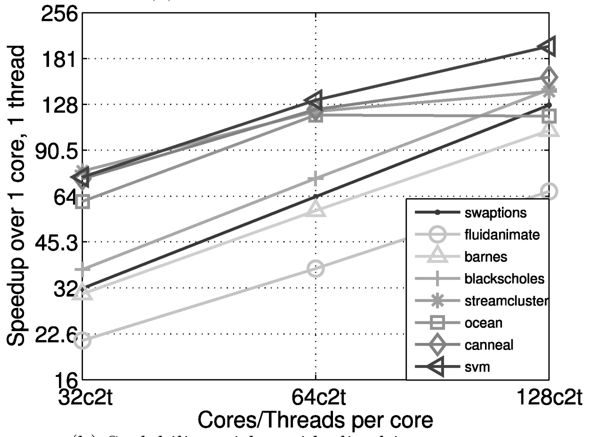
Since we compare routers of different radix, the differences in cycle time may become a concern for routers with more I/Os. However, we find that this is not the case. Increasing the radix primarily affects crossbar size and delay. Using ITRS predictions, for the default 18B flit width and a 128nm wire pitch, a mesh router needs a crossbar of $147 \times 147\mu m^2$, while our highest-radix router (FBFly, 128 cores) requires a $258 \times 258\mu m^2$ crossbar. These are small enough to not require repeaters, and have traversal delays of 50 and 112ps, respectively (derived from ITRS wire delay data for this pitch). While crossbar delay increases, the 62ps difference would be only 12% of the cycle time at 2GHz. Thus, a larger radix imposes a minor timing overhead for the routers we study.

4.3 Workloads

We focus on the scientific and engineering workloads presented in Figure 3(a), with two applications from the SPLASH-2 suite [Woo et al. 1995], five from the recently released PARSEC suite with recognition, mining, and synthesis benchmarks [Bienia et al. 2008]; and one from the BioParallel suite, which focuses on bioinformatics applications [Jaleel et al. 2006]. We simulate the entire parallel section of each application. This set of benchmarks was chosen for its diversity: The applications represent a wide set of domains, with varying behaviors in terms of working set size, amount of data sharing and exchange, synchronization granularity, and synchronization primitives used. Figure 3(b) displays their scalability, measured by varying the number of tiles in the CMP with an idealized interconnect, in systems from 32 to 128 cores (i.e., 64 to 256 threads). For the input sets used, most of the applications scale reasonably well up to 128 cores. Table VII summarizes their most important characteristics as measured on the baseline CMP with 64 cores.

	Input set	**Suite**	**Domain**
swaptions	ns512-sm1000	PARSEC	Financial analysis
fluidanimate	simlarge	PARSEC	Fluid animation
barnes	8,096 bodies	SPLASH2	Barnes-Hut n-body
blackscholes	simlarge	PARSEC	Financial analysis
streamcluster	simlarge	PARSEC	Online clustering
ocean	514×514 grid	SPLASH2	Ocean current sim
canneal	native	PARSEC	Circuit routing
svm	ovarian	BioParallel	Gene classification

(a) Workloads and input sets.



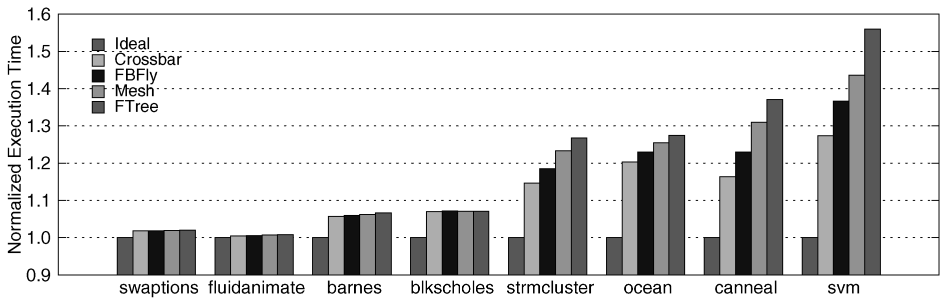
(b) Scalability with an idealized interconnect.

Fig. 3. Workloads, input sets, and scalability with an idealized interconnect.

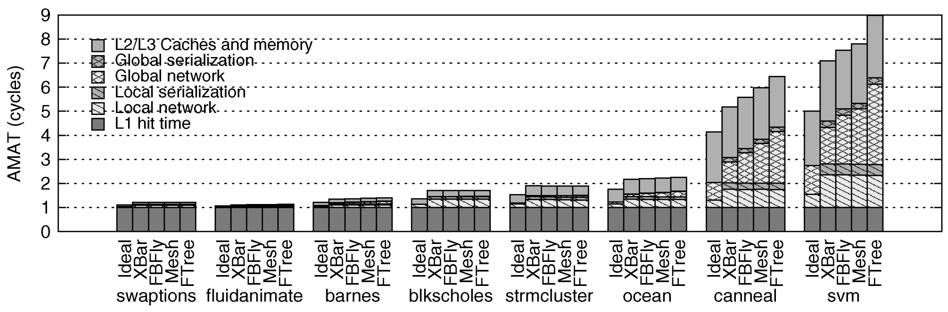
Table VII. Main Workload Characteristics for a 64-core, 128-thread CMP with an Idealized Interconnect

	Instrs.	Loads	Stores	L1D hit rate
swaptions	5.6B	54%	19%	99.3%
fluidanimate	25.9B	39%	12%	99.7%
barnes	1.4B	51%	31%	99.1%
blackscholes	3.3B	34%	21%	97.8%
streamcluster	27.8B	41%	1.2%	97.4%
ocean	2.7B	63%	14%	97.6%
canneal	988M	36%	13%	94.8%
svm	4.4B	52%	0.6%	90.8%

	Misses served by			
	Local L2	Remote L2	L3	Memory
swaptions	99.4%	0.1%	0.0%	0.5%
fluidanimate	78.4%	18.3%	2.6%	0.6%
barnes	77.8%	9.0%	13.1%	0.0%
blackscholes	99.6%	0.4%	0.0%	0.0%
streamcluster	95.7%	4.2%	0.0%	0.0%
ocean	85.8%	9.8%	2.9%	1.4%
canneal	36.4%	26.7%	25.1%	11.7%
svm	48.8%	41.5%	9.7%	0.0%



(a) Execution times relative to the idealized network.



(b) Average memory access time.

Fig. 4. Performance and AMAT for the baseline 64-core CMP, using 18-byte flits.

5. EVALUATION

This section evaluates the design space of CMP interconnects, examining their performance, area, and power requirements. We also study the importance of latency and bandwidth, measure the impact of different L2 cache configurations when the interconnect is taken into account, and the implications of multithreading.

5.1 Baseline Performance

Figure 4(a) shows the relative performance of the different workloads when running on the baseline 64-core, 128-thread CMP. For the three topologies under evaluation (*FBFly*, *Mesh*, and *FTree*), we use 18-byte flits and realistic link latencies determined by wire length. We also present results for two nonrealistic interconnects: *Ideal*, an idealized network with all-to-all communication and no contention or serialization, and *Crossbar*, which has an unrealistic full crossbar with single-cycle links as a global interconnect but illustrates the effects of serialization and contention.

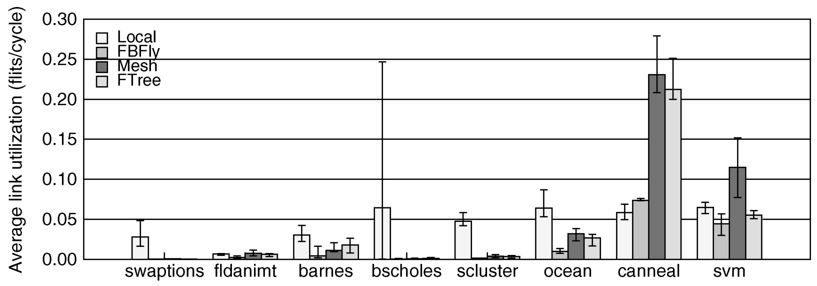
From Figure 4(a), we can clearly distinguish three kinds of applications. Applications such as swaptions and fluidanimate are barely affected by the interconnect. They have high L1 hit rates and do not suffer a noticeable slowdown with respect to the idealized interconnect, as they rarely use the global and local interconnects. Applications such as barnes and blackscholes are affected by the local interconnect only: Their working sets do not fit in the L1 cache, so the number of L1 misses is significant. Since their working sets fit in the L2 and they use coarse-grain synchronization between threads, they rarely use the global interconnect and are barely affected by its organization. Finally, applications such as ocean, streamcluster, and especially canneal and svm are affected by both local and global interconnects. They have moderate to high L1 and L2 miss rates and use the global interconnect frequently. These applications are significantly affected by the choice of global interconnect.

Both the local and the global interconnect choices have a noticeable effect on performance. For applications that stress the global interconnect, the realistic topologies lead to a 17% to 56% performance loss compared to the ideal topology. The flattened butterfly consistently outperforms the other topologies. For svm, the flattened butterfly is 6% and 17% faster than the mesh and the fat tree, respectively. As we explain further in Section 5.2, the performance differences are primarily due to the latency of traversing each topology. Figure 4(b) shows the average memory access time (AMAT) for the different applications and topologies, broken into its memory hierarchy components (L1, L2, L3, directory, and main memory latencies) and interconnect components (local and global network and serialization latencies). The local interconnect typically contributes to the AMAT more than the global interconnect as most of the traffic is between the L1 and L2 caches. We can also see how network latencies dominate over the latencies of the memory hierarchy. In fact, from Figure 4(b), we see that the interconnect is responsible for 60% to 75% of the miss latency.

Comparing Figures 4(a) and 4(b), we see that, although differences in execution time and AMAT are certainly correlated, there can be a significant variation in runtime with very similar AMATs. For example, streamcluster has a 12% runtime variation between topologies with only a 0.7% variation in AMAT. A similar trend can be seen for ocean. This mainly happens because synchronization is slower in networks with higher latency, but slower synchronization leads to higher hit rates (e.g., more time spent spinning on a TTS lock), lowering the AMAT. This shows that execution time or other direct performance metrics should always be used to compare different interconnects instead of lower-level metrics, such as AMAT or miss latency.

5.2 Bandwidth and Latency

Network behavior is characterized by two interacting factors, throughput and latency. Figures 4(a) and 4(b) establish that for applications with high L2 miss rates, global interconnect latency is directly correlated to overall application performance. We first turn our attention to throughput. If the interconnect has significant throughput limitations, we will see large variations in end-to-end latency. Otherwise, latency will be close to zero-load latency, which is determined by the number of hops, the length of links, the serialization latency, and the latency of routers. Figure 5(a) shows average utilization of the network links for the baseline CMP. The plotted ranges indicate the maximum and minimum link utilizations for each network. Global network link utilization is always fairly low, topping at just 23% for the mesh on canneal,



(a) Average link utilization.

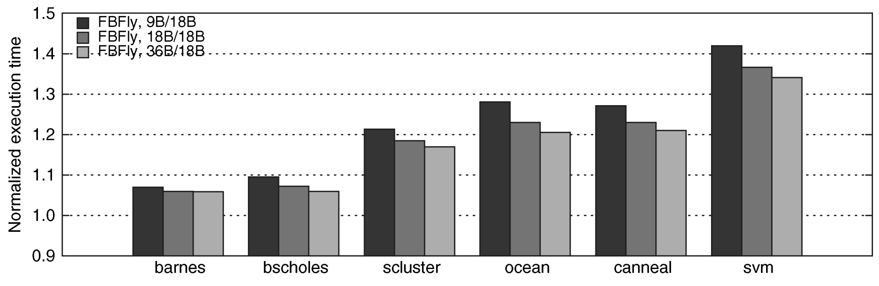
	Average Network Latency	**Blocking Latency (avg/stdev)**
Local	7.75	1.01/1.68
FBFly	15.8	0.31/0.95
Mesh	20.7	1.17/2.22
FTree	27.0	1.56/2.51

(b) Packet latencies for canneal.

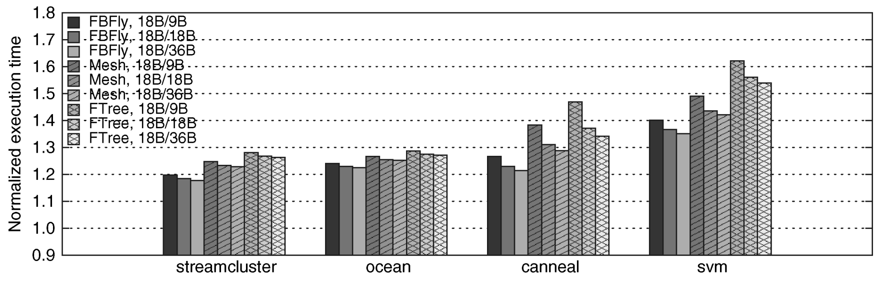
Fig. 5. Link utilization and latencies on a 64-core CMP. The bars indicate the average utilization across links, while the ranges comprise the maximum and minimum individual link utilizations.

and never exceeding 7% for the flattened butterfly. Local network utilization is also generally low. The differences in utilization across links are generally small, with the flattened butterfly having the smallest spreads, and the mesh having the largest ones (due to higher traffic near the center of the chip). Figure 5(b) shows packet latencies for canneal, the application that stresses the interconnects the most. Blocking latency is small, with low spreads even for these worst cases, so we conclude that network congestion is not a significant issue. Hence, it is not worth focusing on adaptive routing algorithms for such systems, and most importantly, throughput metrics are of secondary importance. Instead, the focus should be on latency metrics and optimization techniques.

Given the low link utilization, one may consider to use narrow links that offer lower peak throughput but have reduced power and area overheads. Nevertheless, the link width affects latency as well, as it determines the serialization latency. Figure 6 shows how link width in the local and global interconnects interacts with overall application performance (we only show applications that are affected by varying the flit size). The flit size matters more in the local interconnect, where we see performance dropping by 4% to 11% as we reduce flit size from 36 to 9 bytes. This is partly because the local interconnect has more traffic than the global one (L1 miss rates are higher than L2 miss rates), but mostly because packet serialization is a bigger portion of the overall latency in the smaller local interconnect. In the global interconnect, flit size matters less, with canneal and svm being the only applications that exhibit significant sensitivity.



(a) Performance with varying local flit size.



(b) Performance with varying global flit size.

Fig. 6. Performance for the baseline 64-core CMP with varying: (a) local flit sizes and (b) global flit sizes. The nonvarying flit size is kept to 18 bytes.

5.3 Scaling

The performance impact of the interconnect may vary significantly with the size of the CMP. To quantify this, we scale the number of tiles in the system, while keeping the tile configuration the same. We simulate CMPs with 8, 16, or 32 four-core tiles, corresponding to CMPs with 32 to 128 cores (and 64 to 256 threads). Note that aggregate L2 capacity, L3 capacity, and memory channels grow as we increase the number of tiles.

Figure 7 shows the differences in runtime as we increase the number of tiles. For each processor count, we normalize the execution time with each topology to the execution time with the same processor count and the idealized topology. There are two interesting trends. First, the impact of the interconnect on performance increases with the number of cores. For streamcluster, for example, using the mesh causes a slowdown of 26% and 52% compared to the ideal interconnect for the 64-core and 128-core CMPs, respectively. Second, differences between global interconnect topologies increase with core count. With 64 cores, for instance, streamcluster is 7% faster when using a flattened butterfly instead of a fat tree, while for 128 cores, it is 15% faster. Both effects are more pronounced in the applications that stress the global interconnect frequently (svm, canneal, ocean, and streamcluster).

Regarding the performance scalability of the topologies, it is clear that the mesh is the least scalable, the fat tree shows a slightly better scalability, and

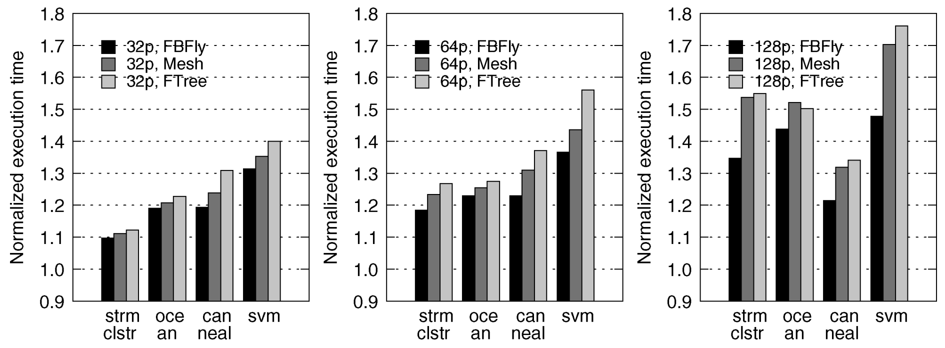
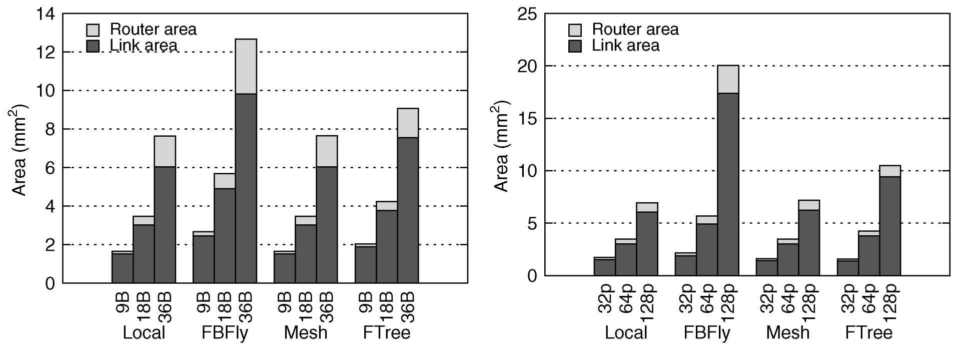


Fig. 7. Execution time normalized for the ideal network, for 32, 64, and 128 cores.



(a) Total interconnect area for a 64-core CMP with varying flit size (9B, 18B, 36B).

(b) Total interconnect area for 18 B flit size with varying number of cores (32–128).

Fig. 8. Area estimations in 32nm for local and global interconnects.

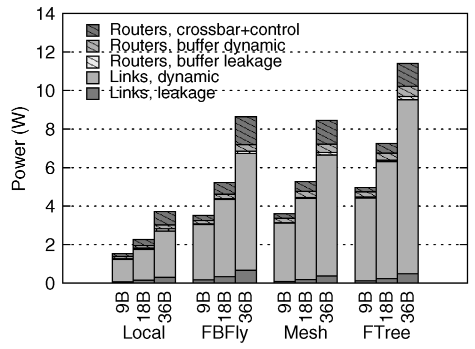
the flattened butterfly is the most scalable. Hence, if area and power budgets allow it, a flattened butterfly seems to be the network of choice.

5.4 Area Comparison

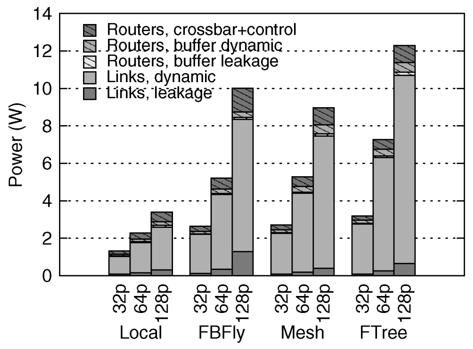
Figure 8 shows the area estimations for the three topologies in 32nm as we vary the flit size (Figure 8(a)) and number of cores (Figure 8(b)). Area is broken down into link and router area. Link area dominates. As a reminder, link area includes repeaters and flip-flops, but not the area of wires as we assumed they are routed over other logic.

Focusing on Figure 8(a), we can see that, for a 18B flit size, the areas of the local and global interconnects are similar in case of the mesh and fat tree (3.5mm^2), and larger for the flattened butterfly (5.6mm^2). Nevertheless, the flattened butterfly with 9-byte flits outperforms the other two networks with 18-byte flits due to the lower overall latencies (see Figure 6(b)) and has smaller area (2.4mm^2).

Looking at Figure 8(b), we can see that the areas of the local interconnect, the mesh and fat tree topologies scale well. However, the flattened butterfly suffers a large area increase for 128 cores. This is due to the higher number



(a) Total interconnect power for a 64-core CMP with varying flit size (9 B, 18 B, 36 B).



(b) Total interconnect power for 18 B flit size with varying number of cores (32–128).

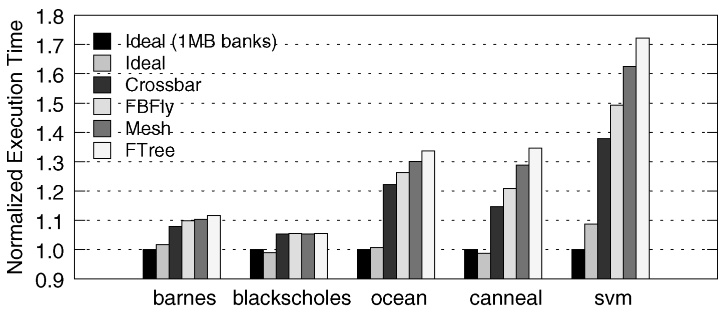
Fig. 9. Interconnect power estimations in 32nm for local and global interconnects when running canneal.

of longer wires and the larger radix of the global routers, since the amount of routers is doubled, and each router must be connected to 10 other global routers. We could improve the scalability of the flattened butterfly by increasing the concentration factor or the dimension of the network [Kim et al. 2007]. This could also be mitigated by limiting the number of links on each dimension to, for example, 4, but would come at the expense of extra delay.

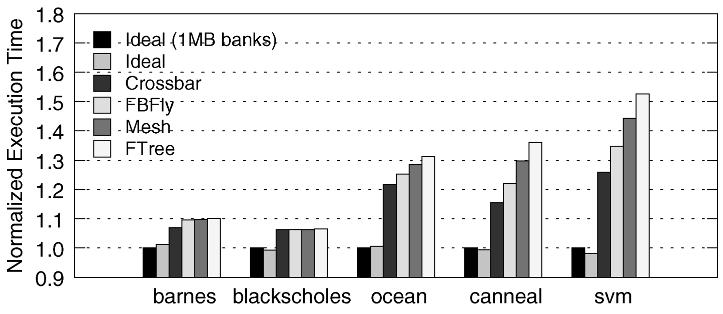
In absolute terms, for the system sizes we consider, the area taken by the interconnect is particularly small with respect to other components of the chip. For example, an 18B flattened butterfly would take 1.6% of the total 64-core chip area. Therefore, even with the large-area flattened butterfly, the overall overhead remains small. These breakdowns do not include wiring area because we assume that wires can be routed over other logic. However, the wiring areas are relatively small, for example, 10.2mm^2 for the mesh and 37mm^2 for the flattened butterfly in the default 64-core CMP. If, for example, the logic density below interconnect wiring decreased by 20%, this would cause additional small overheads of 2 and 7.4mm^2, respectively (corresponding to 0.6% and 2.1% of chip area). Thus, even with wiring area, the interconnect is still a small portion of the overall area.

5.5 Power Comparison

Figure 9 shows the power consumption of the local and global interconnects for canneal, the benchmark that uses them most frequently. Power varies with both flit size and system size. Figure 9(a) shows that the local network power is significantly smaller than the global ones. This happens because links, which consume most of the power, are shorter in the local interconnect, and the number of hops is smaller. For the global interconnects, the flattened butterfly, which is the topology with the largest occupied area, consumes only slightly more power than the mesh due to the higher leakage of the extra links. On the other hand, the fat tree consumes the most power because of the large number of high-radix router hops and link stages that a flit traverses, on average, on the fat tree. Figure 9(b) shows interconnect power for an 18-byte flit



(a) Execution times relative to idealized network, 256KB L2 banks.



(b) Execution times relative to idealized network, 512KB L2 banks.

Fig. 10. Performance of a 64-core CMP, using 18B flits and L2 banks of 256KB and 512KB.

size and a varying number of cores. We observe how all the global topologies scale roughly linearly with system size. Also, note that as the number of cores increases, the flattened butterfly becomes more power hungry because of its superlinear increase in leakage power.

When compared with the total estimated power of the system, we see that the interconnect contributes a moderate amount. For example, when using 18-byte flits on a 64-core system with a flattened butterfly topology, the interconnect consumes 5.2W, roughly 75% of the L2 cache power and a small fraction of the power consumed by the 64 cores.

5.6 Sensitivity to L2 Cache Size

In our evaluation so far, we have used 1MB L2 cache banks. This choice balances the area between cores and caches, but leads to larger L2 caches compared to contemporary designs. In Figure 10, we show the sensitivity of our results to the L2 cache size. We present normalized execution times using 256KB and 512KB L2 banks. Smaller banks lead to higher miss rates but are faster to access (4-cycle tag and 9-cycle data latencies for 512KB, have 4-cycle tag and 8-cycles data for 256KB). Results are presented only for applications with significant L1 misses and are normalized to the configuration with 1MB banks.

Overall, most applications are fairly insensitive to L2 size variations. Only barnes, canneal, and svm exhibit a significant increase of capacity misses with

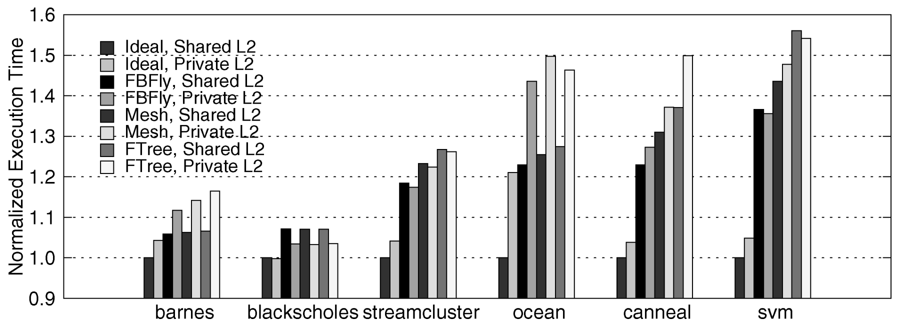


Fig. 11. Execution time with shared and private L2s.

smaller caches. These extra misses do not affect performance in barnes or ocean because multithreading hides their effect, but they significantly degrade the performance of svm.

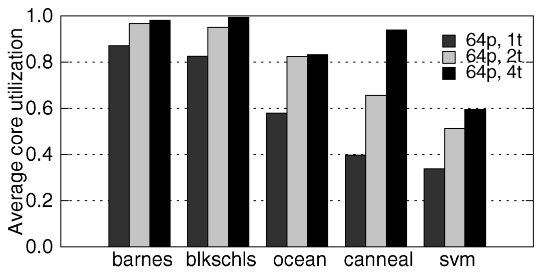
5.7 Sensitivity to L2 Cache Sharing

We evaluate how performance differs if each core has a private L2 cache. Private L2s eliminate the need for the local interconnect, a major contributor to the L2 access latency (see Section 5.1). In the private L2 configurations, each core has a dedicated 3-cycle link between the L1 and L2. On the other hand, sharing the L2 banks has two potential performance benefits: increased L2 hit rates due to better space utilization and faster communication between cores of the same tile, as coherence misses are served by the L2. Of course, sharing L2 banks can also lead to destructive interference between threads that causes more traffic on the global network. We did not notice significant interference issues for the applications we studied.

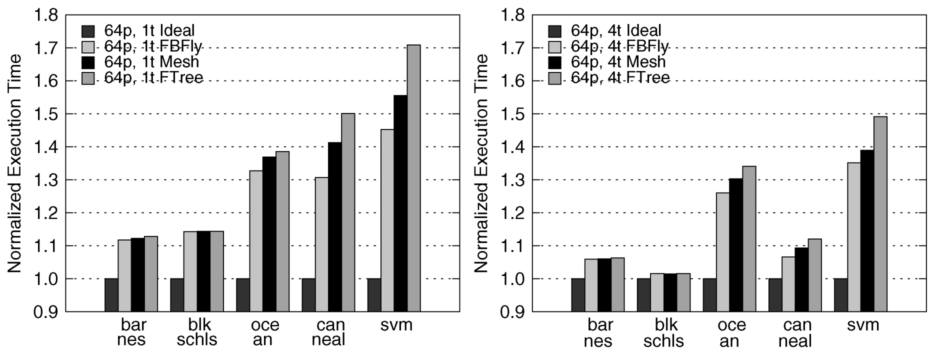
Figure 11 shows how performance varies with shared (per-tile) and private L2s for the different topologies. With an idealized network, all the applications suffer a slowdown with private caches. However, when a realistic interconnect is introduced, the trends change. Blackscholes and streamcluster, which do not stress L2 capacity and have small or moderate interthread communication, benefit from the reduced L2 hit time. For canneal and svm, which have a large amount of communication and sharing, the advantages and disadvantages of L2 sharing roughly cancel out, although the higher amount of global traffic penalizes the slower mesh and fat tree interconnects. Finally, barnes and ocean stress L2 capacity, and private L2s cause a drop in L2 hit rate and are worse than using shared L2s for all the global networks. There are two takeaway points. First, any study of cache sharing schemes should carefully model local and global interconnects. Second, the choice of sharing does not seem to affect the comparison between the three topologies we study.

5.8 Sensitivity to Degree of Multithreading

Multithreading allows us to tolerate access latency in an energy efficient way compared to out-of-order execution techniques. We now explore the trade-offs in varying the degree of multithreading, and explain why 2-way multithreading



(a) Core utilization for different 1, 2, or 4 threads/core, with an idealized network.



(b) Execution times relative to idealized network, for single-threaded and 4-way multithreaded cores.

Fig. 12. Performance characteristics of a 64-core CMP with different degrees of multithreading.

is a reasonable default choice for this study. Figure 12(a) shows the average utilization of the cores when we use 1-, 2-, and 4-way multithreaded cores on our 64-core system, and Figure 12(b) shows the execution time differences between topologies for different degrees of multithreading.

As Figure 12(a) shows, single-threaded cores work well for applications that do not stress the memory hierarchy, but fail to keep the cores highly utilized for memory-intensive applications. Two-way multithreading provides a significant boost for these applications. Going to 4-way multithreading rarely achieves a significant increase in utilization. Figure 12(b) makes clear that having a higher degree of multithreading reduces the relative differences between network topologies, although significant differences remain. However, going beyond 2-way multithreading is rarely useful for these applications, as the overhead of synchronization and load imbalance increase with the number of threads. Only blackscholes and svm exhibit small performance increases of around 3% when going from 2- to 4-way multithreaded cores (and 128 to 256 threads).

Differences in the traffic characteristics of the network are still small, and we have only observed slight increases in interconnect pressure for 4-way multithreading: The average link utilization is maximized by canneal at 26% on a mesh, just a 3% increase from the 2-way multithreaded cores. Thus, our main conclusion still holds: Interconnect throughput is secondary to latency. Although the effect of latency is partially reduced with a high degree of multithreading, these configurations often degrade performance in applications that do not scale linearly with the number of threads.

6. LESSONS AND LIMITATIONS

The following are the major insights and lessons from our study on interconnects for large-scale CMPs:

Overall relevance. The interconnect has a major impact in performance, representing 60% to 75% of the miss latency in all our applications. Moreover, the global interconnect becomes increasingly critical for performance as system size grows. Assuming that the number of cores continues to increase according to Moore's Law and that data-intensive applications become the dominant workloads, the interconnection network will require a major design focus in future CMP architectures.

Bandwidth versus latency. Performance of both local and global interconnects is mainly constrained by latency, not throughput. This is true even for multithreaded cores and applications with large miss rates. Therefore, minimizing latency should be a priority on interconnects for such CMPs.

Topology comparison. In terms of performance and performance-cost, the flattened butterfly topology is the best interconnect choice among the evaluated ones, for the systems we study. It consistently outperforms the mesh and the fat tree, due to its reduced latency, for a marginal increase in system area and power. Even with a limited interconnect area and power budgets, a flattened butterfly with a narrower flit size still outperforms a wider mesh or fat tree and has lower cost. Other topologies with direct links between distant routers, such as express cubes [Grot et al. 2009; Dally 1991] or other low-diameter networks [Xu et al. 2009], are likely to have similar advantages as the flattened butterfly, as they focus on reducing latency. Given this result, we also see no need for interconnect schemes that allow for a reconfigurable topology on these systems [Kim et al. 2008].

Scaling up. All the topologies have reasonable costs for the sizes we explore. Both the mesh and fat tree scale well in terms of area and power. The flattened butterfly would suffer from excessive area and router radix for significantly larger systems. These problems could be solved by using flattened butterflies with a higher concentration factor or dimension (e.g., 3D instead of 2D) [Kim et al. 2007].

Interactions with the memory hierarchy. We have observed significant interactions between the cache hierarchy choices and the design parameters of the interconnection network in terms of traffic patterns and latencies. Idealisms on the one side may skew the results or hide important challenges on the other. Future research for large-scale CMPs should carefully model the interconnection network and cache hierarchy.

Limitations. Despite our efforts to perform a comprehensive exploration, it is important to recognize that our study has limitations. First, we use benchmarks from the engineering and scientific domains that, while diverse, cannot capture all possible behaviors. It would be interesting to repeat this study with memory-intensive commercial workloads and draw conclusions for the server domain. Second, we model a homogeneous CMP with in-order cores, which is a popular option but not the only one. Moreover, while we have made reasonable assumptions on what constitute realistic design points, the design space offers a huge number of alternative choices. Finally, while we attempt to model state-of-the-art interconnects, there are several alternative topologies, flow control, wiring, and router architecture techniques that could be explored. Despite these limitations, we believe that the conclusions of this study are important for understanding the significance of interconnection networks in large-scale CMPs and motivating future research.

7. RELATED WORK

Research in on-chip interconnects has now focused on scalable packet-switched networks [Dally and Towles 2001; De Micheli and Benini 2002; Owens et al. 2007]. Balfour and Dally [2006] studied performance and power trade-offs for tiled CMP interconnects. Using synthetic traces, they conclude that a concentrated mesh topology performs best. Kim et al. [2007] introduced the flattened butterfly topology and demonstrated its advantages with synthetic patterns and source-destination traces. While these studies simulate large-scale interconnects, they neglect the interactions with memory hierarchy revealed with execution-driven simulation. Particular hierarchical network topologies have already been proposed such as meshes [Das et al. 2009]. Recent work has also focused on the implementation of various aspects of the interconnect such as pipelined and speculative routers [Mullins et al. 2004], power-efficient routers [Wang et al. 2003], wiring schemes [Balasubramonian et al. 2005], interconnect-aware coherence protocols [Cheng et al. 2006; Eisley et al. 2006], token flow control [Kumar et al. 2008], predictive routers [Matsutani et al. 2009], and express virtual channels [Kumar et al. 2008].

In the memory hierarchy side, the increasing relevance of wire delay has sparked a significant amount of work in static and dynamic nonuniform caches (NUCA) [Kim et al. 2002; Huh et al. 2005], which can significantly decrease overall access time. In the context of CMPs, Beckmann and Wood [2004] show that the block migration scheme used in dynamic NUCA is inefficient at handling shared data. To cope with this, multiple proposals have used the concept of block replication [Zhang and Asanovic 2005; Chang and Sohi 2006; Beckmann et al. 2006]. All these studies were typically done in the context of small-scale CMPs (with at most 8 or 16 cores). In our study, we consider simple and relatively small L2 caches shared by a few cores, leaving proposals that use replication as the object of future work.

Previous work has demonstrated that the interconnect and the memory hierarchy should be considered together. Kumar et al. [2005] show that when using a bus-based interconnect with point-to point links, a careful codesign of the on-chip network and the cache hierarchy is required to design a balanced, small-scale CMP. They advocate using private L2 caches due to the high area overhead required for shared L2s. Muralimanohar et al. [2008] and Jin et al. [2007] consider interconnect design for large on-chip caches and introduce interconnect enhancements that significantly reduce their overall access latency.

Our work extends past work by performing comprehensive evaluations to characterize the impact of interconnect and cache hierarchy design choices in large-scale CMPs using chip-level metrics. Through these evaluations, we provide guidance to future work by identifying which parameters have the largest overall impact on performance, area, and power, and under which conditions.

8. CONCLUSIONS

We have studied the architecture-level implications of interconnect design for CMPs with up to 128 cores. We adopt a tiled CMP design approach, with separate local (intratile) and global (intertile) interconnects. We have shown that the interconnect is a major component in memory hierarchy and overall performance. Latency, not throughput, is the main interconnect performance constraint for these systems. From the topologies we study, we conclude that the flattened butterfly outperforms the conventional mesh and fat tree, mainly due to its reduced network latency. In terms of cost, all topologies have moderate area and power requirements for the sizes we explore, but they have significant differences in scalability. We have also shown that the global interconnect has a more pronounced impact on performance as the number of cores increases and that the interconnect is sensitive to changes in the cache hierarchy. Therefore, architects must consider and balance both interconnects and cache hierarchies in order to design efficient large-scale CMPs.

Our work indicates that future research on large-scale CMPs should carefully consider the on-chip interconnect along the other components of the memory hierarchy. As we enter the many-core era, the interconnection network will become a major performance bottleneck and further work is required to enhance the scalability of these systems as well as to characterize their limits. Research on interconnects should be more concerned with techniques to reduce latency (e.g., by reducing router delay or number of hops) or mitigate its effect than with schemes that improve the maximum network throughput. A topology that offers significant latency reduction compared to alternative ones is likely to be optimal across all workloads.

ACKNOWLEDGMENTS

We sincerely thank Woongki Baek, Hari Kannan, Jacob Leverich, and the anonymous reviewers for their useful feedback on earlier versions of this manuscript.

REFERENCES

Agarwal, N., Peh, L.-S., and Jha, N. 2007. Garnet: A detailed interconnection network model inside a full-system simulation framework. Tech. rep., Princeton University.

Agarwal, V., Hrishikesh, M., Keckler, S., and Burger, D. 2000. Clock rate versus IPC: The end of the road for conventional microarchitectures. In *Proceedings of the 27th Annual International Symposium on Computer Architecture*. ACM, New York.

Alameldeen, A. R. and Wood, D. A. 2003. Variability in architectural simulations of multi-threaded workloads. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA.

BALASUBRAMONIAN, R., MURALIMANOHAR, N., RAMANI, K., AND VENKATACHALAPATHY, V. 2005. Microarchitectural wire management for performance and power in partitioned architectures. In *Proceedings of the 11th International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA.

BALFOUR, J. AND DALLY, W. J. 2006. Design tradeoffs for tiled CMP on-chip networks. In *Proceedings of the 20th Annual International Conference on Super-Computing*. ACM, New York.

BECKMANN, B. M., MARTY, M. R., AND WOOD, D. A. 2006. ASR: Adaptive selective replication for CMP caches. In *Proceedings of the 39th Annual International Symposium on Microarchitecture.* IEEE, Los Alamitos, CA.

BECKMANN, B. M. AND WOOD, D. A. 2004. Managing wire delay in large chip-multiprocessor caches. In *Proceedings of the 37th annual International Symposium on Microarchitecture.* IEEE, Los Alamitos, CA.

BELL, S., EDWARDS, B., AMANN, J., CONLIN, R., JOYCE, K., LEUNG, V., MACKAY, J., REIF, M., BAO, L., ET AL. 2008. TILE64 processor: A 64-core SoC with mesh interconnect. In *Proceedings of the International Solid-State Circuits Conference.* IEEE, Los Alamitos, CA.

BENKART, P., KAISER, A., MUNDING, A., BSCHORR, M., PFLEIDERER, H.-J., KOHN, E., HEITTMANN, A., HUEBNER, H., AND RAMACHER, U. 2005. 3D chip stack technology using through-chip interconnects. *IEEE Des. Test Comput. 22*, 6, 512–518.

BIENIA, C., KUMAR, S., SINGH, J. P., AND LI, K. 2008. The PARSEC benchmark suite: Characterization and architectural implications. Tech. rep. TR-811-08, Princeton University.

BJERREGAARD, T. AND MAHADEVAN, S. 2006. A survey of research and practices of network-on-chip. *ACM Comput. Surv*. 38, 1.

BONONI, L., CONCER, N., GRAMMATIKAKIS, M., COPPOLA, M., AND LOCATELLI, R. 2007. NoC topologies exploration based on mapping and simulation models. In *Proceedings of the 10th Conference on Digital System Design Architectures, Methods and Tools*. IEEE, Los Alamitos, CA.

CHANG, J. AND SOHI, G. S. 2006. Cooperative caching for chip multiprocessors. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*. ACM, New York.

CHENG, L., MURALIMANOHAR, N., RAMANI, K., BALASUBRAMONIAN, R., AND CARTER, J. B. 2006. Interconnect-aware coherence protocols for chip multiprocessors. In *Proceedings of the 33rd Anuual International Symposium on Computer Architecture.* ACM, New York.

DALLY, W. 1991. Express cubes: Improving the performance of k-ary n-cube interconnection networks. *IEEE Trans. Comput. 40,* 9, 1016–1023.

DALLY, W. J. 1990. Virtual-channel flow control. In *Proceedings of the 17th annual International Symposium on Computer Architecture.* ACM, New York.

DALLY, W. J. AND TOWLES, B. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Conference on Design Automation*. ACM, New York.

DAS, R., EACHEMPATI, S., MISHRA, A. K., NARAYANAN, V., AND DAS, C. R. 2009. Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs. In *Proceedings of the 15th International Symposium on High-Performance Computer Architecture.* IEEE, Los Alamitos, CA.

DE MICHELI, G. AND BENINI, L. 2002. Networks on chip: A new paradigm for systems on chip design. In *Proceedings of the Conference on Design, Automation and Test in Europe.* ACM, New York.

DUATO, J. 1993. A new theory of deadlock-free adaptive multicast routing in wormhole networks. In *Proceedings of the 5th Symposium on Parallel and Distributed Processing.* IEEE, Los Alamitos, CA.

EISLEY, N., PEH, L.-S., AND SHANG, L. 2006. In-network cache coherence. In *Proceedings of the 39th Annual International Symposium on Microarchitecture.* IEEE, Los Alamitos, CA.

GALLES, M. 1997. Spider: A high-speed network interconnect. *IEEE Micro 17*, 1.

GEROSA, G., CURTIS, S., D'ADDEO, M., JIANG, B., KUTTANNA, B., MERCHANT, F., PATEL, B., TAUFIQUE, M., AND SAMARCHI, H. 2008. A sub-1W to 2W low-power IA processor for mobile internet devices and ultra-mobile PCs in 45nm hi-K metal gate CMOS. In *Proceedings of the International Solid-State Circuits Conference.* IEEE, Los Alamitos, CA.

GROT, B., HESTNESS, J., KECKLER, S. W., AND MUTLU, O. 2009. Express cube topologies for on-chip interconnects. In *Proceedings of the 15th International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA.

Ho, R., Mai, K., and Horowitz, M. 2001. The future of wires. *Proc. IEEE. 89*, 4, 24.

Huh, J., Kim, C., Shafi, H., Zhang, L., Burger, D., and Keckler, S. W. 2005. A NUCA substrate for flexible CMP cache sharing. In *Proceedings of the 19th Annual International Conference on Super-Computing.* ACM, New York.

Intel. 2008. Intel Tera-scale Computing Research Program. http://www.intel.com/go/terascale.

ITRS. 2007. International technology roadmap for semiconductors. http://www.itrs.net.

Jaleel, A., Mattina, M., and Jacob, B. 2006. Last level cache performance of data mining workloads on a CMP. In *Proceedings of the 12th International Symposium on High-Performance Computer Architecture.* IEEE, Los Alamitos, CA.

Jin, Y., Kim, E. J., and Yum, K. H. 2007. A domain-specific on-chip network design for large scale cache systems. In *Proceedings of the 13th International on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA.

Jouppi, N. P. 1990. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proceedings of the 17th Annual International Symposium on Computer Architecture.* ACM, New York.

Kahng, A., Li, B., Peh, L.-S., and Samadi, K. 2009. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proceedings of the Conference on Design, Automation and Test in Europe*. ACM, New York.

Kim, C., Burger, D., and Keckler, S. W. 2002. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems.* ACM, New York.

Kim, J., Balfour, J., and Dally, W. 2007. Flattened butterfly topology for on-chip networks. In *Proceedings of the 40th Annual International Symposium on Microarchitecture.* IEEE, Los Alamitos, CA.

Kim, J., Park, D., Theocharides, T., Vijaykrishnan, N., and Das, C. R. 2005. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of the 42nd Annual Conference on Design Automation.* ACM, New York.

Kim, M. M., Davis, J. D., Oskin, M., and Austin, T. 2008. Polymorphic on-chip networks. In *Proceedings of the 35th Annual International Symposium on Computer Architecture.* ACM, New York.

Kumar, A., Peh, L.-S., and Jha, N. K. 2008. Token flow control. In *Proceedings of the 41th Annual International Symposium on Microarchitecture.* IEEE, Los Alamitos, CA.

Kumar, A., Peh, L.-S., Kundu, P., and Jha, N. K. 2008. Toward ideal on-chip communication using express virtual channels. *IEEE Micro. 28*, 1.

Kumar, R., Zyuban, V., and Tullsen, D. M. 2005. Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture.* ACM, New York.

Leiserson, C. E. 1985. Fat-trees: Universal networks for hardware-efficient super-computing. *IEEE Trans. Comput. 34*, 10, 892–901.

Martin, M. M., Sorin, D. J., Beckmann, B. M., Marty, M. R., Xu, M., Alameldeen, A. R., Moore, K. E., Hill, M. D., and Wood, D. A. 2005. Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset. *Comput. Archit. News 33*, 4, 92–99.

Matsutani, H., Koibuchi, M., Amano, H., and Yoshinaga, T. 2009. Prediction router: Yet another low latency on-chip router architecture. In *Proceedings of the 15th International Symposium on High-Performance Computer Architecture.* IEEE, Los Alamitos, CA.

Mullins, R., West, A., and Moore, S. 2004. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*. ACM, New York.

Muralimanohar, N., Balasubramonian, R., and Jouppi, N. P. 2008. Architecting efficient interconnects for large caches with CACTI 6.0. *IEEE Micro. 28*, 1, 69–79.

Nawathe, U. 2007. Design and implementation of Sun's Niagara2 processor. Tech. rep., Sun Microsystems.

Owens, J. D., Dally, W. J., Ho, R., Jayasimha, D. N., Keckler, S. W., and Peh, L.-S. 2007. Research challenges for on-chip interconnection networks. *IEEE Micro. 27*, 5, 96–108.

Thoziyoor, S., Muralimanohar, N., Ahn, J. H., and Jouppi, N. P. 2008. CACTI 5.1. Tech.rep. HPL-2008-20, HP Labs.

Tota, S., Casu, M. R., and Macchiarulo, L. 2006. Implementation analysis of NoC: a MPSoC trace-driven approach. In *Proceedings of the 16th Great Lakes Symposium on VLSI.* ACM, New York.

Tremblay, M. and Chaudhry, S. 2008. A third-generation 65nm 16-core 32-thread plus 32-scout-thread CMT SPARC processor. In *Proceedings of the International Solid-State Circuits Conference.* IEEE, Los Alamitos, CA.

Wang, H., Peh, L.-S., and Malik, S. 2003. Power-driven design of router microarchitectures in on-chip networks. In *Proceedings of the 36th Annual International Symposium on Microarchitecture.* IEEE, Los Alamitos, CA.

Woo, S. C., Ohara, M., Torrie, E., Singh, J. P., and Gupta, A. 1995. The SPLASH-2 programs: Characterization and methodological considerations. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*. ACM, New York.

Xu, Y., Du, Y., Zhao, B., Zhou, X., Zhang, Y., and Jun, Y. 2009. A low-radix and low-diameter 3D interconnection network design. In *Proceedings of the 13th International Symposium on High-Performance Computer Architecture*. IEEE, Los Alamitos, CA.

Zhang, M. and Asanovic, K. 2005. Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture*. ACM, New York.

Received March 2009; revised September 2009; accepted September 2009