Swizzle-Switch Networks for Many-Core Systems

Korey Sewell, Ronald G. Dreslinski, Thomas Manville, Sudhir Satpathy, Nathaniel Pinckney, Geoffrey Blake, Michael Cieslak, Reetuparna Das, Thomas F. Wenisch, *Member, IEEE*, Dennis Sylvester, *Fellow, IEEE*, David Blaauw, *Fellow, IEEE*, and Trevor Mudge, *Fellow, IEEE*

***Abstract*—This work revisits the design of crossbar and high-radix interconnects in light of advances in circuit and layout techniques that improve crossbar scalability, obviating the need for deep multi-stage networks. We employ a new building block, the *Swizzle-Switch*—an energy- and area-efficient switching element that can readily scale to radix 64—that has recently been validated via silicon test chips in 45 nm technology. We evaluate the *Swizzle-Switch* as both the high-radix building block of a Flattened Butterfly and as a single-stage interconnect, the *Swizzle-Switch Network*. In the process we address the architectural and layout challenges associated with centralized crossbar systems. Compared to a conventional Mesh, the Flattened Butterfly provides a 15% performance improvement with a 2.5× reduction in the standard deviation of on-chip access times. The *Swizzle-Switch Network* achieves further gains, providing a 21% improvement in performance, a 3× reduction in on-chip access variability, a 33% reduction in interconnect power, and a 25% reduction in total system energy while only increasing chip area by 7%. Finally, this paper details a 3-D integrated version of the *Swizzle-Switch Network*, showing up to a 30% gain in performance over the 2-D *Swizzle-Switch Network* for benchmarks sensitive to interconnect latency. One major concern with 3-D designs is thermal dissipation. We show through detailed thermal analysis that with the highly energy-efficient *Swizzle-Switch Network* design that the thermal budget is well within that of passive cooling solutions.**

***Index Terms*—Crossbars, manycore systems, multicore processing, network-on-chip, network topology, on-chip interconnects, parallel architectures.**

I. Introduction

THE EMERGENCE of many-core designs has led to a renewed interest in interconnect techniques because intra-chip communication bottlenecks can compromise performance. Although most commercial multi-core designs have used bus-based communication [3], [24], wire delay and bus contention have hindered the scalability of buses past 8–16 cores [12]. As such, it is clear that bus-based interconnects are not suitable for many-core systems.

Network-on-chip (NoC) designs have been advocated as an alternative to bus-based architectures. NoC systems, such as the Tilera Tile64 [49], utilize a distributed multi-stage interconnect design to avoid the scaling issues of long wires. However, this improved scalability comes at the expense of high variability in memory access latencies as well as increased design complexity to guarantee correctness and fairness (e.g., avoiding deadlock, livelock, starvation, etc.).

Crossbar-based architectures, like those in the Niagara2 [26] and IBM BlueGene/Q [21], can provide the uniform memory access latency that is unachievable in multi-stage NoC systems. Additionally, crossbar systems can potentially provide higher bisection bandwidth and lower complexity solutions for quality-of-service guarantees than NoC designs. Despite these advantages, large crossbars are generally considered infeasible because the area and power of traditional matrix-style crossbars grow quadratically with crossbar radix.

In this paper, we revisit the design of crossbar and high-radix interconnects in light of advances in circuit techniques that significantly improve crossbar scalability. Recent work has demonstrated a new circuit-level building block, the *Swizzle-Switch*, an energy- and area-efficient switching element that improves the scalability of crossbars to a higher radix. In addition it provides multicast capability and *least-recently-granted* priority arbitration. Multicasting is the ability for any input to be connected to multiple outputs which simplifies invalidation message delivery in directory protocols. This multicast ability within a permutation network is known as swizzling in the graphics community, from which our design derives its name. The *Swizzle-Switch* has recently been validated with a silicon test chip in 45 nm technology [43]. For purposes of the analysis in this paper we scale and evaluate the design in 32 nm.

We consider two ways in which *Swizzle-Switches* can be deployed in many-core systems: 1) as a high-radix crossbar within the routers of a conventional multi-hop NoC, and 2) as a central switch for a flat crossbar design. Specifically, we evaluate a *Swizzle-Switch*-based, 64-core Flattened Butterfly [29] system and the *Swizzle-Switch Network* (SSN), a system comprising 64 cores and 32 L2 banks connected through central crossbars. We solve key architectural and layout challenges in both designs. In particular, we develop detailed floorplans for each in 32 nm technology, with SPICE analysis, and contrast both the *Swizzle-Switch*-enhanced flattened butterfly and SSN with the Mesh design.

Our detailed results show that the *Swizzle-Switch*-based Flattened Butterfly improves performance by 15% over the Mesh while achieving a 2.5× reduction in the standard deviation of on-chip access times. The SSN attains even higher performance, providing an average of 21% performance improvement, a 33% reduction in interconnect power and a 25% reduction in total system energy over the Mesh with only a 7% area overhead

Manuscript received December 30, 2011; revised March 11, 2012; accepted March 29, 2012. Date of publication May 17, 2012; date of current version June 07, 2012. The authors would like to thank ARM and the National Science Foundation for supporting this work. This paper was recommended by Guest Editor J. Kim.

The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2012.2193936

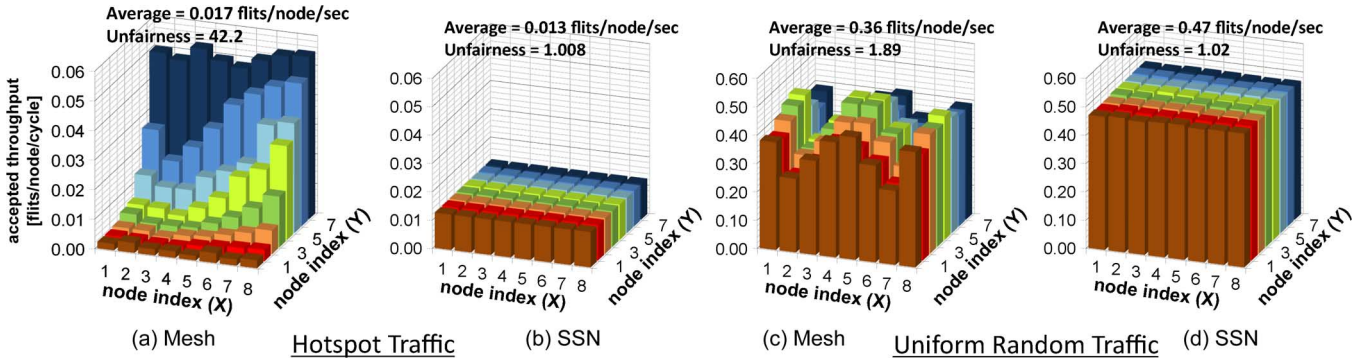


Fig. 1. QoS in network-on-chip. Figures (a) and (b): All nodes generate traffic directed to a hotspot located at (8, 8) with injection rate of 0.05 flit/cycle/node, and the bar graphs show the accepted service rate per source node for (a) 8 × 8 mesh and (b) 64-radix SSN. Figures (c) and (d) demonstrate the same effect for uniform random traffic at injection rate of 1 flit/cycle/node. Unfairness metric derived from [13].

with no global wiring congestion. In addition, the more uniform transfer latency of the flat SSN reduces the standard deviation of on-chip access latencies by 3× with respect to the Mesh.

Finally, the design of the *Swizzle-Switch Network* is investigated in a 3-D integrated design. With the use of through-silicon-via (TSV) technology a 3-D-SSN design approach is explored. In the 3-D-SSN the *Swizzle-Switch* is folded across several layers, reducing the area and capacitance of the *Swizzle-Switch*. This reduced capacitance leads to a increased *Swizzle-Switch* speed. By reducing the number of elements on each layer, the interconnect to and from the *Swizzle-Switch* is also improved. Overall, the interconnect speed is increased by 1.8× in a four-layer system. This increased interconnect performance results in up to a 30% increase in performance for benchmarks that are sensitive to interconnect latency. Additionally, thermal analysis shows that the 3-D-SSN can be cooled using passive cooling solutions.

The rest of the paper is organized as follows. Section III presents the details for our high-radix self-arbitrating crossbar element, the *Swizzle-Switch*. In Section IV, we present the *Swizzle-Switch Network* and contrast it with two NoC topologies: the Mesh and Flattened Butterfly. Section V provides details of our evaluation methodology. We present results in Section VI. Section VII details on potential 3-D implementation of the SSN and presents performance results. Section VIII provides background information on interconnection techniques. Finally, we conclude in Section IX.

II. Motivation

The continued shift toward many-core systems in the architecture community has led to renewed research in interconnects, particularly for on-chip communication. To provide better bandwidth, early multi-core systems transitioned from bus-based interconnect fabrics to crossbars [2], [26], [52]. However, the increased power consumption and die area from high-radix crossbars has led researchers to explore alternatives to flat crossbar topologies [51]. In systems whose core counts approach 64–128, several studies have noted that creating a crossbar to interconnect all cores would cause the interconnect's power and area to dominate the system [32]. Consequently, there has been a paradigm shift towards packet-switched,

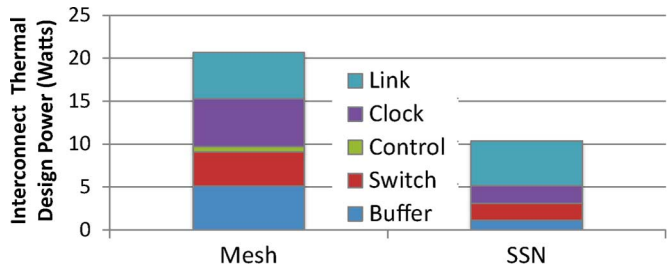


Fig. 2. Comparison of interconnect power for the SSN and mesh.

on-chip networks with regular topologies such as meshes [22], [49] and rings [19], [45].

This shift from a flat interconnect model to nonuniform, multi-hop interconnects has come at the cost of nonuniform cache access (NUCA) latencies [27]. The ability to provide uniform latency makes using a crossbar an appealing option because predictable latencies remove the need for complex techniques for routing algorithms [39], quality-of-service [18], congestion management [33], data placement [35], and thread scheduling [23].

Fig. 1 shows the high variability of multi-hop, on-chip networks compared to the single-hop *Swizzle-Switch Network* (SSN) detailed in Section IV. In this example, both networks are assumed to be run at the same frequency. The Mesh network's accepted throughput at any given node is highly dependent upon the location of the destination node. Under worst-case hotspot traffic, nodes closest to destination ($\mathrm{node}_{8,8}$) receive the highest throughput, while nodes closest to the center (e.g., $\mathrm{node}_{4,4}$) receive the highest throughput when traffic is uniformly distributed. In contrast, the SSN distributes its throughput evenly amongst its nodes allowing for it to see a 40× and 87% fairness improvement for hotspot and uniform random traffic, respectively. There are many research papers that address fairness issues in NoCs. However, these solutions can involve complex mechanisms [18], [33] to enforce fairness whereas crossbar topologies simplifies these quality of service concerns by construction.

Additionally, the reduced wiring complexity of a NoC system is bought at a price other than nonuniform latency: collisions can occur within the network. To resolve these collisions and avoid protocol deadlocks, on-chip networks usually require additional

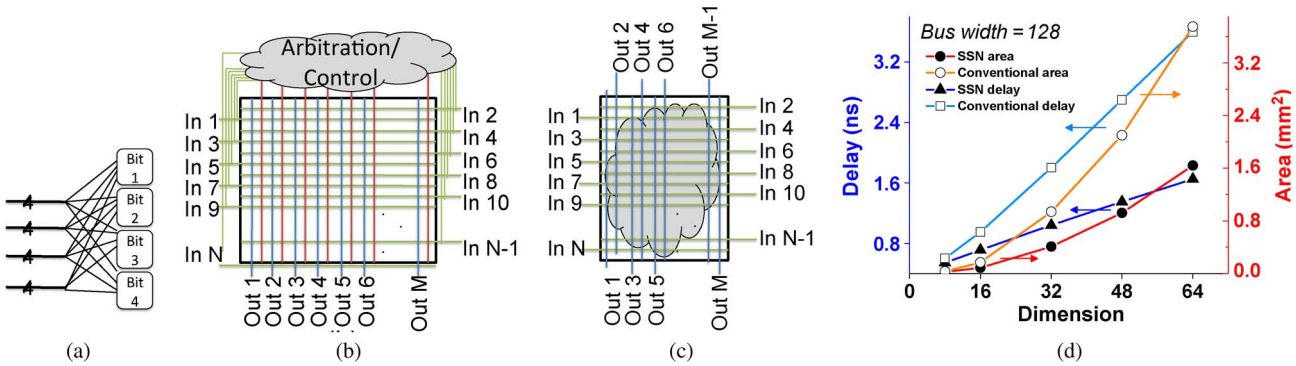


Fig. 3. High level view of (a) bus interleaving required for mux-based crossbars; (b) a traditional matrix style crossbar with arbiter/controller consuming space and requiring additional input wires; (c) the proposed *Swizzle-Switch* design that reuses input/output busses for programming/arbitration of the crossbar with the arbitration logic placed under the dense metal interconnect; and (d) scaling trends of the *Swizzle-Switch* Interconnect in 32 nm versus a conventional crossbar (Simulated).

buffers per router node (e.g., to provide multiple protocol lanes or virtual channels), which consume significant power and area. As a result, Borkar *et al.* [10] predict that many-core NoCs could consume as much as 80 W in future systems. In contrast, a flat crossbar is nonblocking and does not require intermediate buffers, reducing power and area overheads.

Previous studies address the power and latency scalability challenges of NoCs by building concentrated and hierarchical topologies to reduce the required number of on-chip routers [4], [14], [29]. Our proposed SSN realizes a single-router flat interconnect with a scalable high-radix crossbar design and minimal end-point buffering. Fig. 2 illustrates the power savings achieved by our proposed design for a synthetic benchmark designed to saturate the network (details for this analysis are found in Section IV).

In this paper, we study the limits of high-radix crossbar interconnects. We leverage a new circuit element, the *Swizzle-Switch*, a highly scalable and energy efficient crossbar. We enhance the previous *Swizzle-Switch* design and then use it to build the SSN: a MOESI-based, many-core processor. The SSN challenges the conventional notion that crossbars do not scale up to 64 cores by demonstrating how to build a high-radix crossbar in a many-core system.

III. Swizzle-Switch: A High-Radix, Self-Arbitrating Crossbar

The SSN is a high-radix crossbar system built using a number of *Swizzle-Switch* components. Each *Swizzle-Switch* component employs several techniques to reduce the area and power overhead of high-radix crossbars. The underlying circuit technology of the *Swizzle-Switch* was first described in work by Satpathy *et al.* [41]–[43]. These brief circuit-oriented papers describe 65 nm and 45 nm test chips that uses synthetic traffic to measure crossbar power and performance. In this paper we explain for the first time the theory of operation of this new building block and evaluate our designs using multithreaded applications. In this section, we describe the *Swizzle-Switch* and evaluate a scaled design in 32 nm. We address the key architectural challenges of building a MOESI-coherence 64-core chip multiprocessor using *Swizzle-Switches* in Section IV.

A. Overview

Conventional mux-based crossbars suffer from a layout challenge at high bus widths because of complex wire interleaving within the crossbar itself. Fig. 3(a) shows how four buses each with four bits must be interleaved to connect to a mux-based crossbar. To avoid these interleaving structures, more recent crossbars use matrix-style structures.

Fig. 3(b) shows a typical matrix-style crossbar, where the connection to each output is made at a crosspoint inside the crossbar, the inputs can be in any order and no interleaving of bits from buses is required. Conventionally, these matrix-style interconnects consist of a *crossbar that routes data* and a *separate arbiter that configures the crossbar*. This decoupled approach poses two hurdles to scalability: 1) the routing to and from the arbiter becomes more challenging as the number of sources and destinations increase and 2) the arbitration logic grows more complex as the radix of the crossbar increases. Arbiters that need to distribute their arbitration over multiple stages incur the overhead of flip-flops to store the control flow signals. The work done by Passas [38] is an example of the prohibitive overheads that can be seen when implementing a a multistage arbiter a high-radix crossbar. In Passas' work, a radix-128 crossbar that implements the iSlip algorithm [36] is shown to have an crossbar arbiter that consumes 60 % of the total crossbar area. This result further illustrates the area and power problems conventional matrix-style crossbars suffer when scaled to large core counts.

To overcome these limitations, we can use the *Swizzle-Switch* (SS) to replace conventional matrix-style crossbars. Although the SS has the same asymptotic behavior as the matrix-style crossbar, $O(n^2)$, it uses circuit techniques to reduce the multiplicative constants of that behavior to readily scale to at least 64 cores. The SS combines the routing-dominated crossbar and logic-dominated arbiter by *embedding the arbitration logic within the router crosspoints*. Furthermore, it *reuses input/output buses for arbitration*, producing a compact design. Fig. 3(c) shows a high-level *Swizzle-Switch* design. To reduce power, the SS uses SRAM-like technology with low-swing output wires and a single-ended thyristor-based sense amplifier [42]. To understand the scalability of the SS interconnect compared to a conventional crossbar design, we generated a series

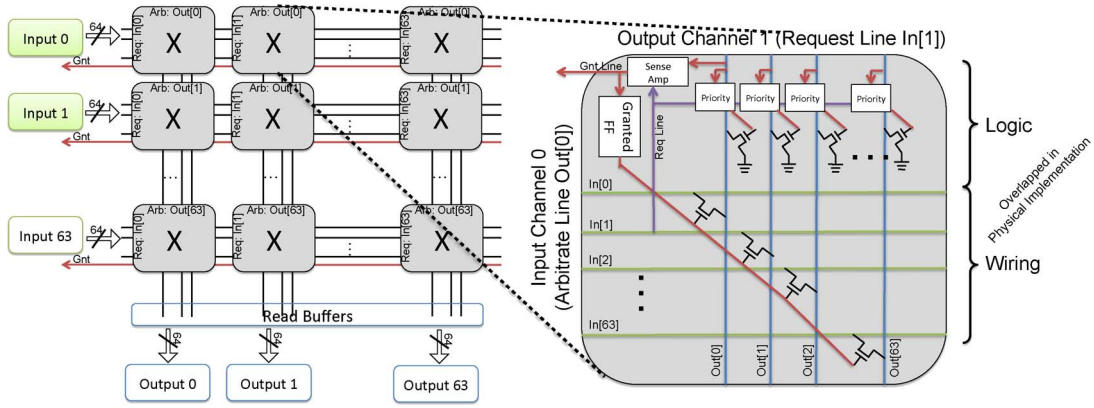


Fig. 4. Circuit implementation of the *Swizzle-Switch* Interconnect. Each output *column* in the interconnect uses the same request bit from each input bus. Each input *row* uses the same bit from each output bus to perform arbitration. The expanded view of the crosspoint shows the stored configuration and crosspoint connections for each bit. It also shows the programming of priority bits using the output bus. Because the crosspoint is for Input Row 0, the arbitration sense amp is on *output wire 0*. Similarly, because it is Output column 1, the request line is drawn from *input wire 1*.

of layouts across a wide range of radices. Fig. 3(d) shows the results of this analysis. The SS design scales far better to higher radix designs, consuming 2.7× less area and performing 2.6× faster at a radix of 64.

Although the *Swizzle-Switch* (SS) can implement arbitration schemes with low logic overhead, we do not propose the SS as a replacement for all existing switch fabrics. Rather, SS can be used to aid the scalability of current state of the art fabrics.

B. Layout and Data-Transmission Phase

As shown in Fig. 4, the *Swizzle-Switch*'s input and output buses run perpendicular, creating a matrix of crosspoints each containing a storage element to designate connectivity. Along a column (output), at most one connection can be made, allowing each output to connect to at most one input. Along a row (input), multiple connections can be made. This allows a single input to multicast to a subset of outputs, or broadcast to all outputs.

Every output channel operates **independently** in one of two modes, *data-transmission* or *arbitration*. When an output channel is not allocated, it enters arbitration mode until an input is granted access to the output channel. Once the output channel has been granted, it transitions to data-transmission mode. Data transmission can continue for several cycles as the input channel transfers the complete payload. Once the input is finished transmitting data, it releases the channel and allows the output channel to move back into arbitration mode.

The circuit details of the data transmission phase are illustrated in Fig. 4. During data transmission, the output buses are pre-charged to "1." The input channel then drives the horizontal wires with the data. At crosspoints where the "Granted FF (Flip Flop)" stores a "1," the input bitlines are coupled to the precharged output bitlines with a pass gate. If the input bitline is "0," the output bitline will discharge and the sense amplifier at the read-buffer will sense the data. The "Granted FF" uses a thyristor-based sense amplifier to set the enabled latch, which only enables the discharge of the output bus for a short period of time, reducing the voltage swing on the output wire. This reduced swing coupled with the single-ended sense amplifier helps to increase the speed, reduce the crosstalk, and reduce the power consumption of the *Swizzle-Switch*.

C. Arbitration Phase

Any output channel that is not in the data transmission phase is in arbitration phase. In this phase, each output channel will grant a single input channel access to transmit data. The input channel with highest priority is granted access. The *Swizzle-Switch* uses an *inhibit-based approach* to accomplish this arbitration. When an output channel is requested, input channels will inhibit other inputs with lower priority. Consequently, managing the priorities of each input allows for the implementation of various priority schemes. There are two novel aspects of the arbitration design. 1) The priority bits stored in each crosspoint are used to determine the winner of the arbitration. 2) Each input repurposes a particular bit of horizontal input bus to assert a request signal and is assigned a particular bit of the vertical output bus to use as an inhibit line.

1) Arbitration Mechanism: The conceptual view of inhibit-based arbitration of a single output column for a five-input *Swizzle-Switch* is shown in Fig. 5(a). The arbitration for an output channel can represented by a matrix (M) of requests (In) and inhibits (X). A row of bits in M correspond to the storage elements labelled "priority bits" in the expanded crosspoint view in Fig. 4. An input In_i inhibits an input In_j if and only if the entry $\mathrm{M}_{(i,j)}$ is 1, indicating that In_i has priority over In_j. In this example, if input In_0 and In_1 are both arbitrating for the output channel then In_0 would inhibit In_1 (since $\mathrm{M}_{(0,1)} = 1$) and win the arbitration between the two requesting inputs. However, if In_0 and In_2 were in arbitration with each other, then In_2 would win the arbitration by inhibiting In_0 (since $\mathrm{M}_{(2,0)} = 1$). The priority for an input is then the number of *other* inputs that it can inhibit.

Fig. 6 illustrates the operation of the arbitration circuit. The same priority scheme from Fig. 5(a) is used. The vertical output $OutK_i$ bit-line is repurposed as inhibit line X_i during the arbitration phase. The input channel In_3 has 1's stored in all its priority bits $\mathrm{M}_{3,j}$ and hence has the highest priority. The input channel In_2 has priority over only inputs In_0 and In_1 (because only these priority bits are set). At the start of arbitration, all inhibit lines are precharged. Then, for each competing

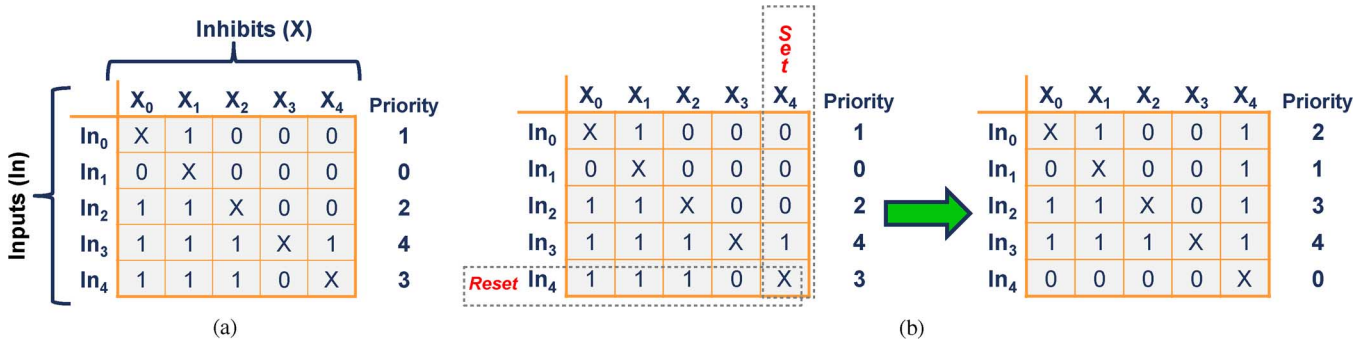


Fig. 5. Conceptual example of (a) *Swizzle-Switch* Arbitration and (b) Least-Recently-Granted priority update for *Swizzle-Switch*. A matrix represents one complete output column. Each output column arbitrates and transmits data independently of other output columns.

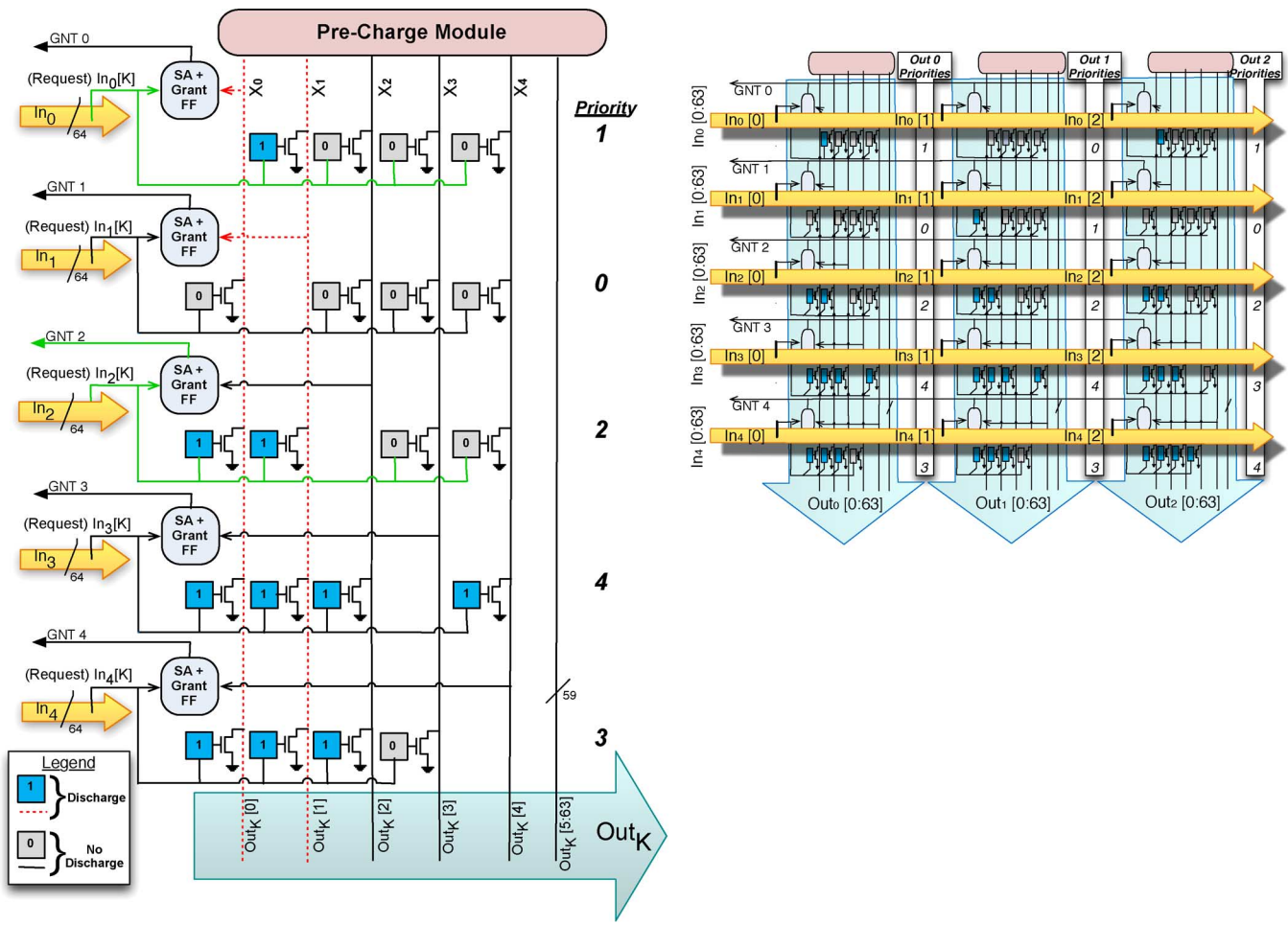


Fig. 6. (Above) a 5 × 3 crossbar showing arbitration circuits. Each output column arbitrates independently and maintains it's own priority. Some outputs may be in arbitration mode while others are transmitting data. (Left) Detailed blowup of arbitration for the Kth output column of a five-input *Swizzle-Switch* interconnect. *In 0* and *In 2* both request the output channel. *In 0* discharges output bit-line 1, and *In 2* discharges both output bit-lines 0 and 1. *In 0*'s sense amp sees the discharged output bit-line 0 and recognizes that it lost the arbitration. *In 2*'s sense amp sees a charged line and hence is granted the output channel. The grant signal is latched and sent back to the input to begin data-transmission mode.

input channel, if the priority bit [i.e., $\mathrm{M}_{(i,j)}$] is set, the corresponding inhibit line (i.e., X_j) is discharged via a pass transistor. Each input channel In_i monitors inhibit line X_i to determine if it won the arbitration. If the inhibit line is discharged, a higher-priority channel must have requested the output and consequently In_i loses the arbitration. Conversely, if the inhibit line remains precharged, then no higher-priority channel requested the output. The arbitration result is latched in the "GrantedFF" to set up the connection for data transmission.

Note that though our examples illustrate unicast requests, each input can request multiple output columns. Together, the bit-lines of an input port constitute a multi-hot signal to request a subset of output channels. The priority bits stored in the different crosspoints are used to determine the winner of the

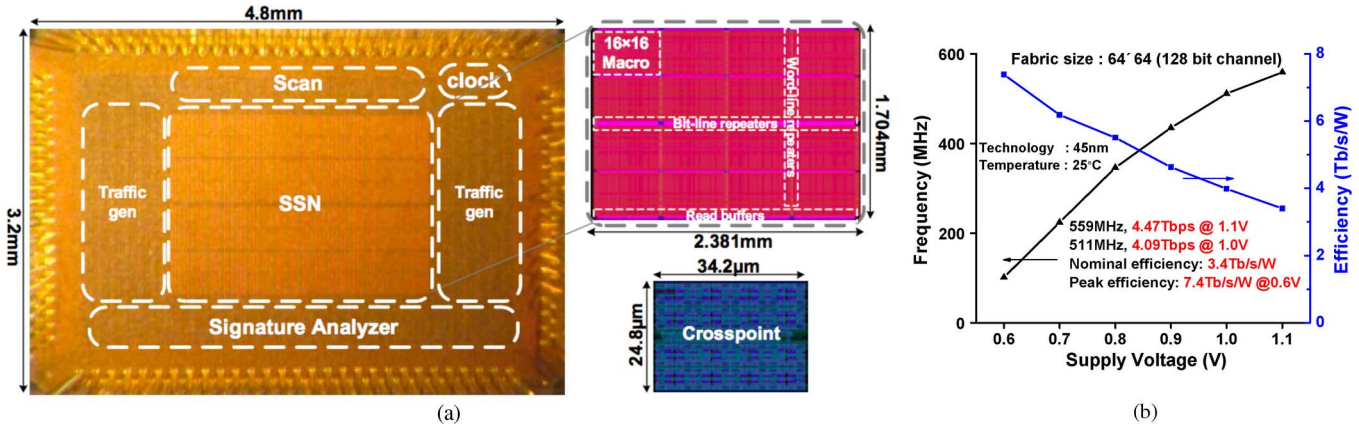


Fig. 7. (a) Die Photo of the 45-nm silicon test chip. (b) Measured frequency and bandwidth efficiency of the silicon test chip.

arbitration. By updating the priority every time the channel is granted, fair arbitration can be achieved.

2) Arbitration Priority Update: Fair scheduling algorithms can be implemented in the *Swizzle-Switch* by resetting and setting the appropriate inhibit bits in the arbitration matrix. Fig. 5(b) shows how *least recently granted* (LRG) priority can be achieved using the *Swizzle-Switch*'s inhibit-based priority scheme. In this example, inputs 0, 2, and 4 are all arbitrating for the output channel. Input 4 wins the arbitration since it has the highest priority amongst the arbitrating requests. To achieve an LRG update, we first reset all bits in the row of In_4 to enforce that input 4 can not inhibit any other request in the matrix. Next, we set all bits in the inhibit column of X_4 to enforce that all requests can inhibit input 4 during the next arbitration cycle. Thus, the combination of set and reset operations achieves LRG by giving least priority to input In_4 and incrementing the priority of all other inputs that previously had lower priority then In_4 by 1. LRG arbitration helps to ensure starvation-free operation.

D. Silicon Validation

Satpathy *et al.* have validated the feasibility of the *Swizzle-Switch* building block with a fabricated and tested silicon prototype [43]. The prototype chip was manufactured in a commercial 45 nm technology and consisted of a 64 × 64 Swizzle interconnect with 128-bit busses. The total size of the Swizzle interconnect was 4 mm^2. The interconnect was driven by synthetic traffic generators (including support for broadcast and multicast traffic) with a built-in test circuit that verified correct transmission. Fig. 7 shows the die photo of the silicon test chip. Measurements of the chip show that at full voltage the Swizzle interconnect operates around 559 MHz and provides 4.47 Tb/s of bandwidth. The total power of the interconnect was 1.32 W at full voltage under a 20% switching factor. Whereas Satpathy *et al.* [43] introduces the *Swizzle-Switch* circuit design, that work does not explain the theory of operation nor address the architectural and layout challenges of how the new crossbar design can be leveraged in practical chip interconnect architectures, which is the focus of Section IV.

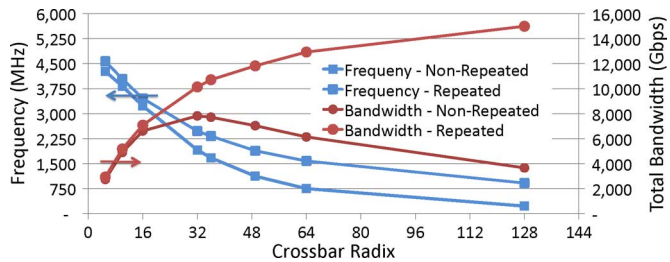


Fig. 8. Bandwidth and Speed of a *Swizzle-Switch* with 128-bit busses in 32 nm.

E. Enhanced 32 nm Design and Analysis

We scale the design of the *Swizzle-Switch* (SS) to 32 nm through SPICE modeling. All results are validated against the 45 nm test chip. In addition, we further optimize the crossbar for higher frequencies at high radices. The schematic-based SPICE crossbar model includes word-/bit-line drivers, parasitic loads, and worst-case coupling capacitance to neighboring wires. For a specific radix and bus width, we optimize driver sizes to maximize crossbar frequency and utilize the additional metallization layers to reduce area. We evaluate two crossbar designs: one with single-segment word-/bit-lines and another with optimally-spaced repeaters. Metal wire delay scales quadratically with distance while repeated wire delay scales linearly. Thus, for high crossbar radices, adding repeaters drastically reduces word- and bit-line delay at the cost of increased power.

The frequency of the SS speed is dominated by wire *RC* delay. The decrease in the length and width of the wires from 45 nm to 32 nm leads to a nearly 2× increase in speed. Futhermore, an enhanced version of the repeater structure as well as improved placement of the repeaters acounts for additional speedup. Fig. 8 shows total crossbar bandwidth and maximum frequency as a function of radix, with and without repeaters, for a design with 128-bit busses. A crossbar of radix 64 × 64 × 128 supports a bandwidth of 13 Tb/s and can operate at greater than 1.5 GHz with an area of about 1 mm^2.

IV. Interconnect Architectures

In this section, we present three different many-core chip designs based on alternative interconnect architectures. First, we present our proposed flat crossbar design, the *Swizzle-Switch*

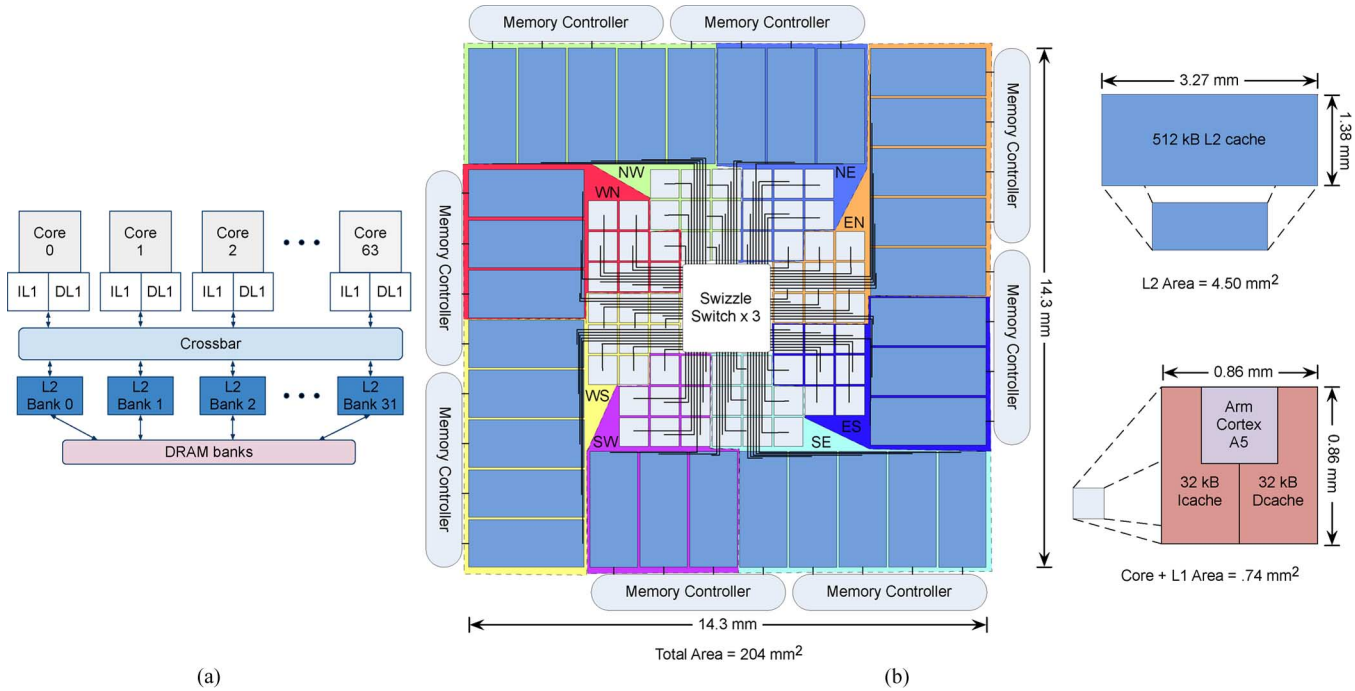


Fig. 9. High-level architecture diagram (a) of a 64-core system built with a *Swizzle-Switch* based crossbar. Floor-plan (b) of our system and estimated dimensions. Octants are colored to aid the reader in seeing how wires leave the crossbar. The total chip area is 204 mm^2, each core/L1 tile consumes 0.74 mm^2, the L2 tiles consume 4.5 mm^2, and the *Swizzle-Switch* consumes 6.65 mm^2.

Network (SSN). The SSN uses insights derived from coherence traffic classification to build a 64-core, cache-coherent many-core architecture—a crossbar design that was previously thought to be impractical [32], [40]. As a baseline for comparison, we describe a conventional Mesh network. Finally, we describe how the high-radix *Swizzle-Switches* can be used to improve a Flattened Butterfly interconnect.

A. Common Components

For all three designs, we target an industrial 32 nm process. We estimate core size, speed, and power based on published characteristics of an ARM Cortex-A5 [3] in 32 nm. The 32 nm A5 achieves a frequency of 1.5 GHz and occupies 0.18 mm^2. We estimate cache area, latencies, and power using Artisan SRAM compiler estimates and SPICE simulations. We select total cache size to target a 200 mm^2 chip in 32 nm technology. The design uses eight interleaved memory controllers. Each L2 address range is assigned to the nearest memory controller in order to minimize interconnect congestion.

Our target 32 nm process provides a nine-layer metallization stack. In this metallization stack there are four $1X$, two $2X$, two $4X$, and one $8X$ metal layers. The $1X$ metal layers and one of the $2X$ metal layers are reserved for local routing (within the core/cache). The $8X$ layer is reserved for power and clock routing. That leaves two $2X$ and two $4X$ layers for global routing. The interconnect for the NoC and SSN uses only parts of one $2X$ and parts of one $4X$ layer. Wire delays were determined using wire models from the design kit using SPICE analysis including repeaters, taking into account cross-coupling capacitance of neighboring wires and metal layers. For interconnect wires, we consider four options that trade off area for speed. We can use a $4X$ or $2X$ metal layer with either single or double spacing. Repeater insertion is adjusted so that repeaters are placed in the gaps between cores. The repeater placement was considered for all topologies to accurately estimate timing. The resulting wire delays ranged from 55–350 ps/mm depending on repeater placement, wire spacing, and metal layer.

B. Swizzle Switch Network

The *Swizzle-Switch Network (SSN)* combines novel circuit and architectural insights to challenge the conventional scaling limitations of crossbars. Crossbar-based systems are desirable in many-core chips because they can ease the burden of managing highly variable memory access latencies. Thus, an SSN-based system can take advantage of new crossbar technology to provide uniform memory access at the many-core level.

Fig. 9(a) shows one possible configuration of a SSN system. The SSN connects 64 cores, each with their own private L1 instruction and data caches, to 32 banks of L2 cache. The L2 cache banks are then connected to DRAM. Fig. 9(b) shows one potential layout of such a system. Each tile comprises an ARM Cortex A5 and 32 kB L1 I/D caches. The L2 cache is banked in 32 tiles of 512 kB each and placed around the perimeter of the cores. Memory controllers are placed in the periphery and are directly connected to the L2s. The colored octants indicate the association of L2 and core tiles to the centralized switch.

1) Coherence Protocol: A novel design aspect of the SSN is it's use of three *Swizzle-Switches* to enable a directory-based, MOESI coherence protocol. MOESI coherence requires an interconnect fabric to facilitate communication among the private (L1s) and shared (L2s) caches in the system. Fig. 10(a) classifies coherence messages into four types: L1 $\rightarrow$ L1, L1 $\rightarrow$ L2,

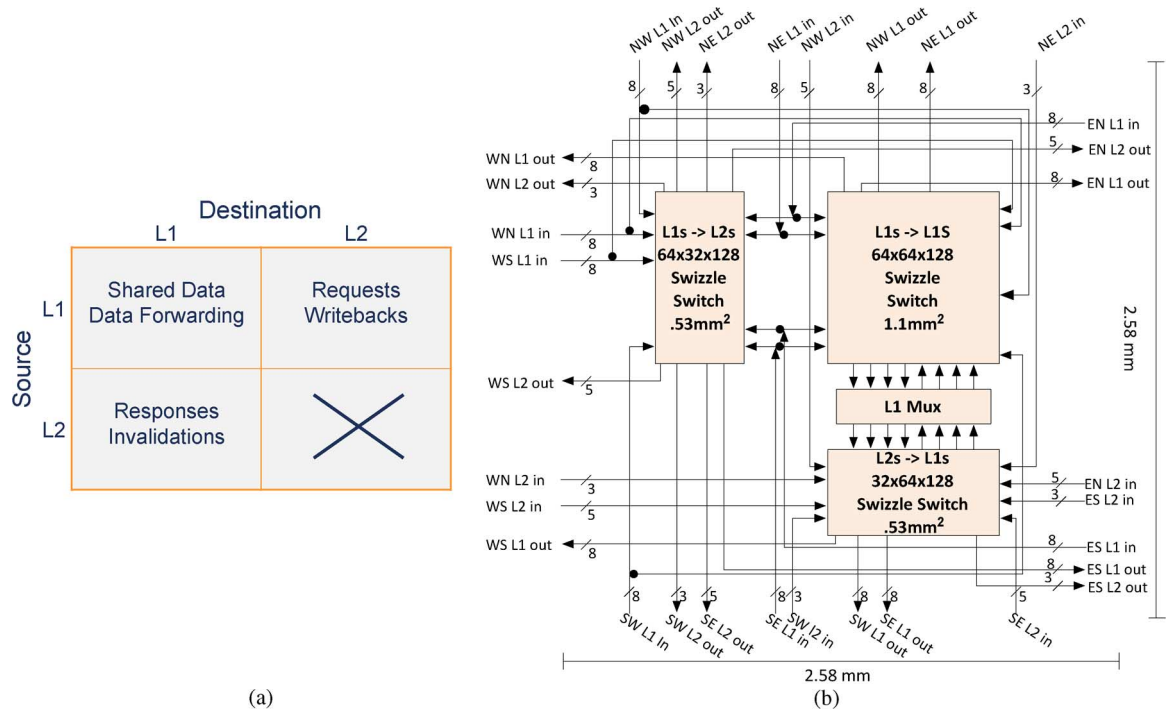


Fig. 10. (a) Classification of communication messages required for coherence. (b) Floor-plan and wiring diagram for combining three *Swizzle-Switches* into a $64 \times 64 \times 128$ bit crossbar. The wires are labeled by the quadrant to which they connect. Each wire in the diagram represents either 3, 5, or 8 busses, where each bus is 128-bits. The overall area of the Crossbar is 6.65 mm^2($\sim$4% of the 64 tile system).

L2 $\rightarrow$ L1, and L2 $\rightarrow$ L2. Note that MOESI protocols require no L2 $\rightarrow$ L2 communication. Consequently, we optimize the SSN to provide only the three required communication paths, each via dedicated *Swizzle-Switches* (omitting a crossbar for L2 $\rightarrow$ L2 communication). This optimization reduces SSN power requirements by 17% relative to a switch with all four communication paths.

In addition, the multicast ability of the SSN facilitates further traffic and power optimizations for invalidation messages. Within the SSN, the invalidations can be multicast to several L1 caches simultaneously (i.e., driving the SSN input bus only once). In contrast, NoC designs either transmit individual invalidation messages per destination or must employ sophisticated control policies to enable broadcast/multicast [25].

2) Timing and Layout Evaluation: Fig. 10(b) illustrates the layout of the three *Swizzle-Switches* needed to build a complete SSN crossbar: two for the bi-directional interconnect from L1s $\rightarrow$ L2s and one for communication of shared data from L1s $\rightarrow$ L1s. Each depicted wire represents several 128-bit buses. Busses are grouped by the chip octant to which they are routed, as identified in Fig. 9. The diagram reflects the relative locations of input/output busses and corresponds to the floorplan in Fig. 9.

To calculate the area of the SSN itself, several measurements are needed. We determine the area of each *Swizzle-Switch* from detailed layouts; the two smaller *Swizzle-Switches* occupy 0.53 mm^2 while the larger requires 1.1 mm^2. Routing over the *Swizzle-Switch* is not possible (as it occupies all global-routing metal layers), so all busses that pass around the *Swizzle-Switches* consume area. We determined the required routing area for these busses as well as area for fan-out to the proper metal layers and spacing for connections to the global interconnect.

Overall, including these route-around overheads, the *Swizzle-Switch* network is 2.58 mm $\times$ 2.58 mm, for a total area of 6.65 mm^2 ($\sim$4% of total chip area). The SSN layout also includes some empty space at the periphery due to symmetry/routing constraints, however, this empty space could be used for other circuitry. Considering this additional overhead, the total area of the SSN-based chip is 204 mm^2, a 7% increase in area over the Mesh and Flattened Butterfly topologies.

Interconnect signals take one cycle to reach the crossbar, one cycle to arbitrate, one cycle to pass data through the crossbar, and one cycle to reach the destination. Global wires are routed in a mixture of $2X$ and $4X$ metal depending on the distance routed. As a result, the longest wires operate at 1.7 GHz. At the most routing dense point in the layout, just outside the Swizzle-Switch, the $4X$ metal layer utilization due to SSN routing is 60% of routing tracks and the $2X$ metal layer utilization is 40% of routing tracks. In other parts of the chip, routing density due to SSN routing drops substantially and is not a significant factor. From Fig. 8 the $64 \times 64 \times 128$ repeated crossbar configuration was selected to maximize bandwidth while achieving a 1.5 GHz clock frequency to match the cores.

3) Reliability: The SSN's monotholic design can potentially make it susceptible to reliability issues if one or more of the input/output ports experiences a fault. Modifications to enhance the robustness of an SSN are available because of it's SRAM-like layout and it's small total area. Error correction techniques (e.g., ECC) that are used in standard SRAM systems can also be used to enhance SSN reliability. Additionally, redundant output ports could be used to aid fault recovery mechanisms.

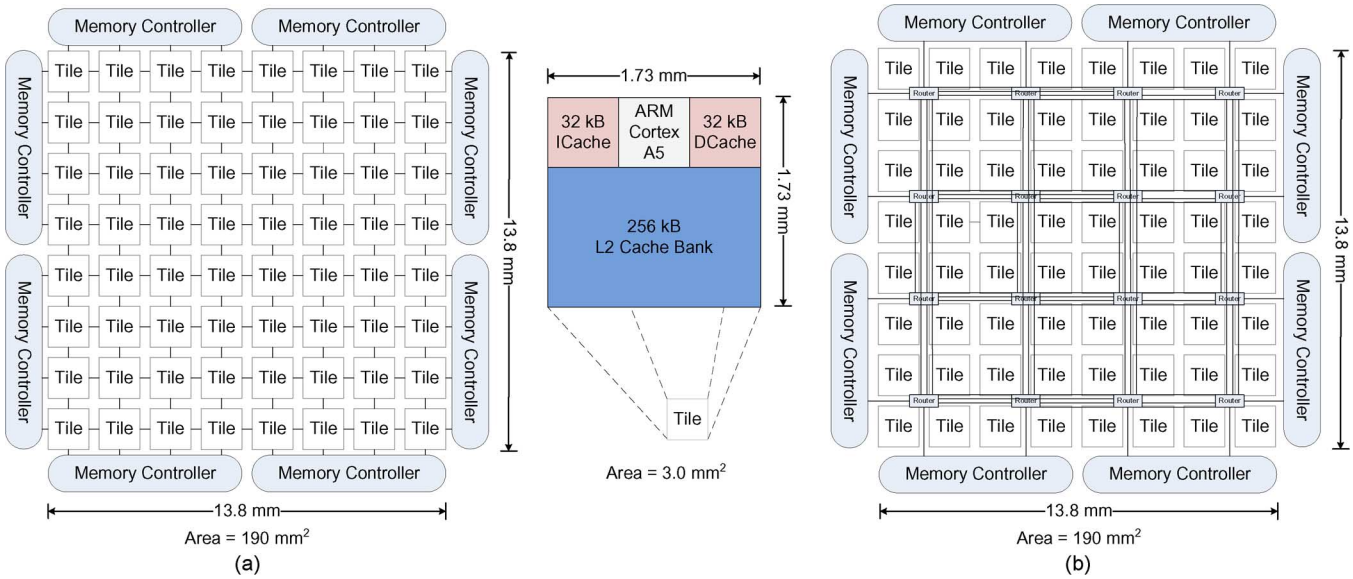


Fig. 11. Floor-plan of the Mesh and Flattened Butterfly systems with estimated dimensions. The total size of both chips is 190 mm^2.

C. Mesh Topology

We contrast the SSN against a Mesh topology, which has been used in a number of recent many-core designs [22], [29]. A Mesh is amenable to a tiled design, is easy to lay out, and does not require any long, cross-chip wires. Moreover, by distributing L2 cache slices in each core tile, Meshes can facilitate L2 cache designs that provide low latency from the core to its associated L2 slice. Despite these advantages, Mesh interconnects do not always scale well because they are vulnerable to issues such as router congestion, network power, and nonuniform (and high worst-case) access latencies to remote L2 banks. In addition, bursty and hotspot traffic patterns that often arise in real applications can lead to high queuing delays even if overall interconnect utilization is relatively low.

Fig. 11(a) shows our Mesh layout. Each tile comprises one ARM Cortex-A5, private 32 kB instruction and data caches, a 256 kB slice of the shared L2 cache, and a router that links the tile to its four nearest neighbors. Tiles at the periphery connect to the memory controllers. We employ four-stage pipelined routers, using lookahead routing to eliminate the need for a route computation stage [16]. To prevent deadlock, we use XY-dimension ordered routing and implement three virtual networks (request, response, writeback) over one physical network. To reduce congestion (e.g., due to head-of-line blocking), we then allocate three virtual channels per virtual network.

The area of each tile is 3 mm^2, resulting in a total chip area of 190 mm^2 (excluding memory controllers). Router latency is dominated by virtual channel allocation and arbitration time, resulting in a peak frequency of $\sim$3.5 GHz. The interconnect links (channels) are 16 bytes wide and 1.73 mm in length. When routed in $2X$ double-space metal they can achieve a speed of 3.1 GHz. We operate the NoC at 3 GHz to match an even multiple of the core frequency.

D. Flattened Butterfly

As a second baseline for comparison, we have also designed a flattened butterfly network (FBFly) [29], as recent work has demonstrated that this topology can outperform meshes due to decreased hop count between any tile pair. For example, a 4-ary, 3-flat FBFly can support 64 cores while *bounding* the router hop count to 2 (as opposed to a 64-core Mesh *average* hop count of 8). The FBFly requires considerably higher radix routers than the Mesh. Large high-radix routers are typically slow, however, *by using a Swizzle-Switch as the crossbar element within the router*, we enable the design of a FBFly at a 40% higher frequency compared to a conventional crossbar-based router design. Thus, we demonstrate how a *Swizzle-Switch* can also be used as a *building block* for high-radix network design.

Fig. 11(b) shows the proposed layout of the Flattened Butterfly topology. Tiles in the FBFly are identical to those in the Mesh. In this layout, each of the 16 routers are connected to 4 tiles creating a 64 node network. The radix for each router in the FBFly can be either 14 or 16 because there are three links for routers in the same row, three links for routers in the same column, four links to the local L1s, four links to the local L2s, and up to two links to memory controllers. Although adaptive routing can take advantage of the FBFly's path diversity, implementing such a technique typically includes a signficant amount of router complexity. This work simplifies FBFly routing by always routing to the XY-dimension ordered path that requires two-hops from source to destination.

The links to nearest-neighbor routers are 3.46 mm and are routed in the $4X$ double spaced metal. These nearest-neighbor links can operate faster than 3 GHz. For links to nonnearest-neighbor routers, the wire lengths are either 6.92 or 10.38 mm. These links are both routed in $4\times$ double spaced metal, resulting in up to 620 ns of delay. To allow the network to operate at 3 GHz, these links are pipelined in two stages.

TABLE I
gem5 64-CORE SIMULATION PARAMETERS

Feature	Mesh & Flattened Butterfly	*Swizzle-Switch Network*
Processors	64 in-order cores, 1 IPC, 1.5 GHz	
L1 Caches	32kB I/D Caches, 4-way associative, 64-byte line size, 1 cycle latency	
L2 Caches	Shared L2, 16 MB, 64-way banked, 8-way associative, 64-byte line size, 10 cycle latency	Shared L2, 16MB, 32-way banked, 16-way associative, 64-byte line size, 11 cycle latency
Interconnect	3.0 GHz, 128-bit channels, 4-stage Routers, 3 Virt. Networks w/3 VCs per Network	1.5 GHz, 64x32x128bit Swizzle-Switch Network
Main Memory	4096MB, 50 cycle latency	

TABLE II
CACHE MISS RATES AND L1 MISS LATENCY (IN CPU CYCLES)

Benchmark	L1 MPKI	L2 MPKI	L1 Miss Latency to an On-Chip† Location						Speedup over Mesh	
			Mesh		Flattened Butterfly		*Swizzle-Switch Network*			
			Average	Std. Dev.	Average	Std. Dev.	Average	Std. Dev.	**FBFly**	**SSN**
Barnes	6.2	0.5	53.6	16.1	32.0	5.5	27.1	2.2	1.12×	1.15×
Cholesky	2.4	1.2	57.2	16.7	32.3	5.8	24.7	5.3	1.04×	1.07×
FFT	4.4	1.4	57.5	16.7	33.2	6.7	27.2	7.3	1.11×	1.14×
FMM	2.5	0.7	55.4	16.5	32.1	5.7	25.9	3.9	1.11×	1.15×
LuContig	1.2	0.7	57.1	16.4	32.0	5.6	23.9	5.3	1.02×	1.03×
LuNonContig	1.8	2.0	60.5	16.1	32.3	7.4	22.1	9.6	1.04×	1.05×
OceanContig	17.9	7.1	54.7	16.2	32.5	6.6	26.4	6.0	1.29×	1.43×
OceanNonContig	27.2	8.2	54.2	16.0	32.3	6.0	26.7	5.1	1.31×	1.45×
Radix	22.1	8.6	54.4	16.1	27.2	10.3	26.8	4.3	1.28×	1.37×
Raytrace	7.7	2.0	56.2	16.7	32.6	6.0	25.9	3.8	1.57×	1.82×
WaterNSquared	3.3	0.5	54.4	16.2	32.4	5.9	26.9	4.3	1.05×	1.07×
WaterSpatial	0.6	0.2	56.9	16.6	32.5	6.8	25.0	7.8	1.02×	1.02×
Average			**56.2**	**16.4**	**31.9**	**6.5**	**25.6**	**5.4**	**1.15×**	**1.21×**

† Excludes main memory accesses

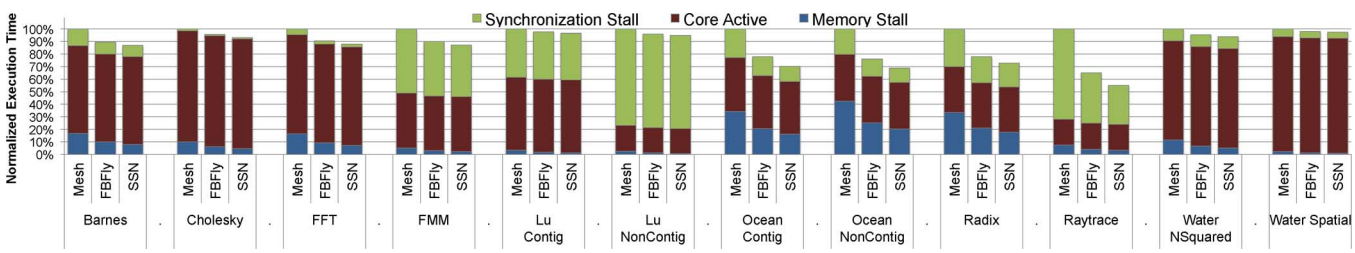


Fig. 12. Cycle Analysis for 64 core Mesh, FBFly, and SSN topologies during parallel regions of the SPLASH2 benchmarks.

V. SIMULATION METHODOLOGY

We evaluate our three interconnect designs using detailed timing simulation with the *gem5* full-system simulator [6]. We extend *gem5* to model the three interconnects and a MOESI directory-based coherence protocol. We configure the interconnect simulation using timing characteristics derived from the SPICE and layout analysis discussed in the previous sections. Table I details the simulation parameters. To account for nondeterminism in threaded workloads, we randomly perturb memory access latencies and run multiple simulations to arrive at stable runtimes as described by Alameldeen *et al.* [1].

We evaluate using benchmarks from the SPLASH2 [50] suite. The SPLASH2 benchmarks are of particular interest for the study of on-chip interconnects as they have diverse sharing and data migration patterns between cores as shown by Barrow-Williams *et al.* [5].

VI. RESULTS

We evaluate the the *Swizzle-Switch Network* (SSN), Mesh, and Flattened Butterfly (FBFly) systems according to four metrics: overall runtime performance, average miss latency, miss latency variation, and energy/power. Our analysis shows that both of the topologies enabled by our high radix crossbar—the FBFly and the SSN—perform noticeably better than the Mesh even though the SSN runs at only half the frequency. The FBFly is 15% faster in overall runtime, has a 1.76× reduction in average L1 on-chip miss latency, and experiences a 2.52× reduction in the standard deviation of L1 on-chip miss latency compared to the Mesh. The SSN has a 33% lower interconnect power, decreases the run time by 25%, reduces the average L1 on-chip miss latency by 2.2×, and provides a 3× reduction in the standard deviation of L1 on-chip miss latency relative to the Mesh.

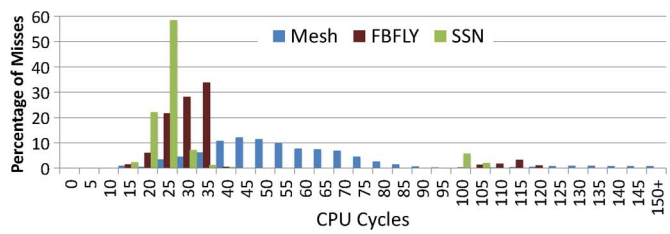


Fig. 13. Histogram of L1 cache miss latency for the Radix benchmark.

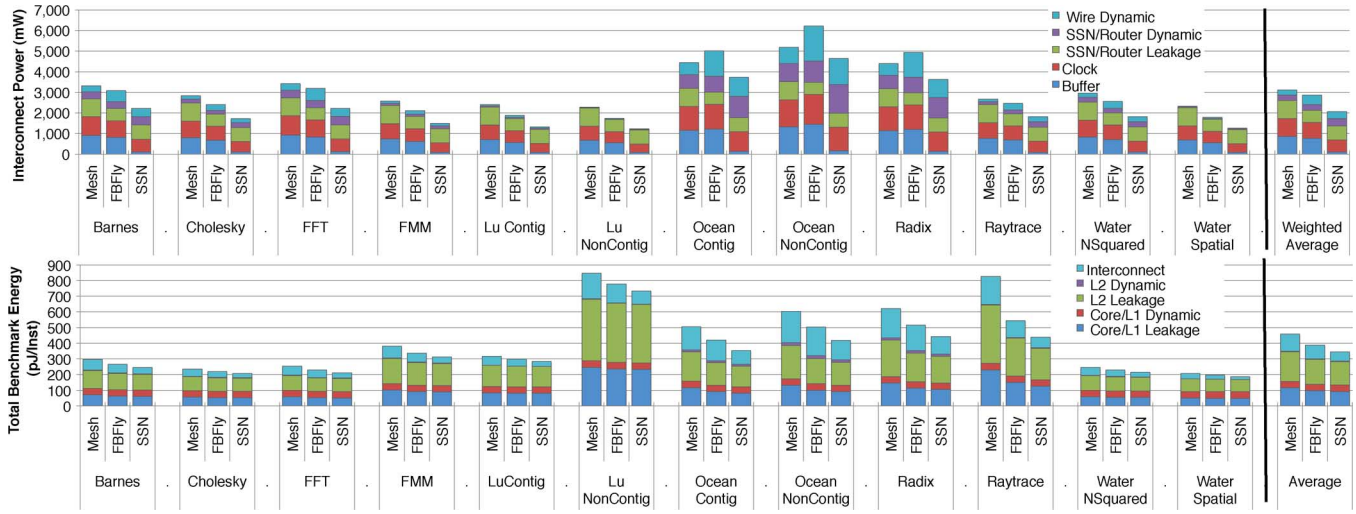


Fig. 14. Total interconnect power (top) and energy (bottom) broken down by components within the Mesh, FBFly, and SSN systems for all benchmarks tested. Overall the SSN reduces interconnect power by 33% over the Mesh and 28% over the FBFly on average. As a result of the lower interconnect power and better performance the total SSN system energy is 25% less than the Mesh and 11% less than the FBFly.

A. Performance Analysis

Table II shows the speedup for each benchmark and Fig. 12 shows execution time breakdowns comprising three categories: core active cycles, memory stall cycles, and synchronization stall cycles. We observe three different performance-impact scenarios in the results. The first arises for benchmarks with high L1 miss rates and substantial sensitivity to L2 access stalls. *OceanContig*, *OceanNonContig*, and *Radix* all have high L1 Misses Per KiloInstruction (MPKI) as shown in Table II and also spend a substantial fraction of execution time on memory stalls as shown in Fig. 12. The *Swizzle-Switch*-based topologies substantially accelerate these workloads, due to the improved average L2 access latency.

The second class of workloads, including *Raytrace* and *FMM*, spend a large fraction of time in synchronization stalls. These particular benchmarks have locks that are sensitive to miss latency. As average miss latency improves, synchronization time is also reduced, yielding significant speedups. When synchronization stalls arise due to load imbalance, as in *LuNonContig*, there is no significant speedup since improving memory latency does not resolve the load imbalance.

The last scenario arises for benchmarks with a low L1 MPKI, for example, *WaterSpatial*. Such benchmarks are insensitive to L2 latency as their working sets fit in L1, and thus, achieve only minimal performance gains (2%) from the faster interconnects.

In Fig. 13 we show the miss latency distribution for the Radix benchmark (other benchmarks have similar miss latency distributions). Within this distribution, accesses with latencies from 10–80 cycles are serviced on chip, whereas those with latencies above 100 cycles are misses to main memory. The figure illustrates that the high-radix interconnects achieve tighter latency distributions for on-chip accesses. The wide latency variance in the Mesh is due to the highly-variable hop count (as many as eight hops) for some messages. In contrast, messages on the FBFly require at most two hops, while the SSN requires only one (variance arises only due to endpoint queueing and latency differences between L1-to-L1 and L2-to-L1 transfers). The average and standard deviation of the miss latency is given in Table II. While the Mesh maintains a standard deviation of L1 miss latency of 16.4 cycles, the FBFly is able to achieve a standard deviation of 6.5 cycles and the SSN tightens the standard deviation even further to 5.4 cycles. The more predictable access latency in the FBFly and SSN interconnects makes it easier for programmers to analyze performance and balance work across cores and reduces the need for careful on-chip data placement.

Overall, we have shown that high-radix topologies enabled by the *Swizzle-Switch* scale well to 64 cores and achieve significant speedups over a Mesh. The FBFly on average sees a 15% speedup over the Mesh, while the SSN further increases the average overall speedup over the Mesh to 21%.

B. Energy and Power Analysis

Previously, one of the main criticisms of high radix crossbars was their high power consumption [32]. However, our optimized design demonstrates that high-radix interconnects can be more power efficient than low-radix topologies, which require a larger number of routers and buffers. Fig. 14 shows the power consumption for the three interconnects broken down into switch, buffers, link, and clocking power subcomponents. The Mesh has the highest interconnect power consumption. The FBFly, on the other hand, runs at the same high frequency as the Mesh and has pipeline buffers on its longer wires, therefore requiring higher dynamic wire power than the Mesh but has lower overall router power consumption. The SSN trades off reduction in buffer power with increase in wire dynamic power while maintaining similar switch power. Overall, the SSN reduces interconnect power by 33% over the Mesh and 28% over the FBFly on average.

The total system energy consumption of the SSN is reduced because of the overall runtime improvement. The energy savings arises from conserving leakage energy in the core and L2, which shrink as performance increases and total runtime is reduced. From the plot one can see that the largest improvements

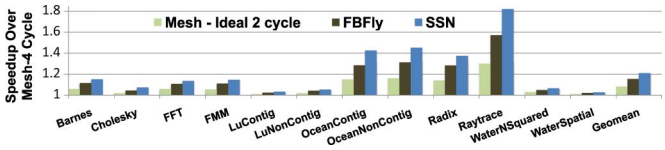


Fig. 15. SSN, FBFly, and two-cycle Mesh speedups over a four-cycle Mesh.

in energy consumption correlate to the benchmarks with the largest performance improvement, i.e., *Raytrace*. As a result of the lower interconnect power and reduced runtime, the total SSN system energy is 25% less than the Mesh and 11% less than the FBFly.

C. Sensitivity Analysis

To further explore SSN tradeoffs, we perform sensitivity analysis on three key parameters: router pipeline depth, interconnect frequency, and traffic from out-of-order cores.

1) Router Pipelines: Fig. 15 shows the performance gains of an idealized two-stage router pipeline (employing pipeline bypassing and speculative virtual channel allocation) [39] over the baseline four-stage Mesh routers. We study an idealized design that assumes bypassing is always possible and speculative virtual channel allocation never fails (i.e., by configuring the simulation with a two-stage pipeline). Even under these idealized assumptions, the SSN outperforms the two-stage Mesh by an average of 12.4%. Note that this estimate is conservative: gains would be higher when compared to an accurate implementation of a speculative router (due to mispeculation stalls). We do not consider speculative routers for the FBFly topology because the higher radix make their implementation considerably more difficult.

2) Interconnect Frequency: Fig. 16 evaluates the performance of the SSN, the four-cycle Mesh and the ideal two-cycle Mesh when the interconnect frequency is varied from 1.5 to 6.0 GHz. To isolate the impact of the interconnect, the cores held at a constant 1.5 GHz for all design points. As expected, a flat crossbar system is always advantageous to a Mesh when the interconnect frequencies are equal. However, the study shows that the average number of router hops in the network-on-chip system is such a prohibitive factor that it would take a Mesh running at 6.0 GHz to match the performance of a SSN running at 1.5 GHz. Consequently, we show that the benefit of building a flat system can be significant given the feasibility constraints of building a network-on-chip with an aggressive clock and a small number of pipeline stages.

3) Out-of-Order Cores: We demonsrate the performance impact of using out-of-order (O3) cores on a 64-core system in Fig. 17. Replacing the in-order cores with O3 cores would present significant area and power overheads to both the SSN and NoC systems. We ignore those limitations in this example and focus on generating as much core-stimulated traffic as possible. We do this by placing 1.5 GHz, eight-wide O3 cores (64 Inst. Window, 64 LD/ST Queue, and 256 Phys. Regs.) in each network and assuming the frequency of the networks are equivalent to their achieved speeds when using in-order cores.

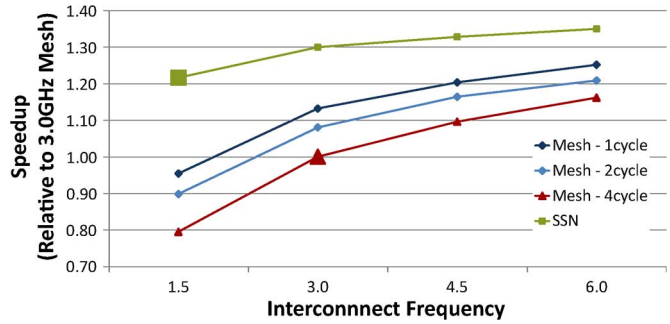


Fig. 16. Sensitivity to the interconnect frequency for the Mesh and SSN (Cores remain at 1.5 GHz). The enlarged data points for the SSN and Mesh represent the configurations used in Section VI-A.

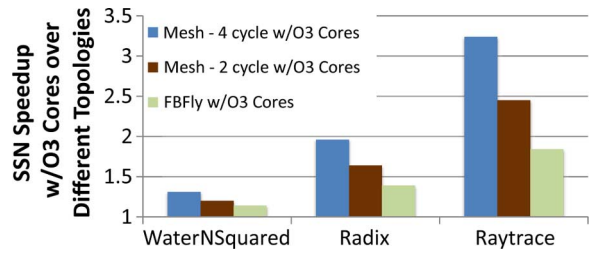


Fig. 17. Speedups of a 64-core SSN using Out-of-Order (O3) cores over 64-core NoCs also using O3 cores. Benchmarks shown represent the three traffic classes referenced in Section VI-A.

The increased traffic from O3 cores magnifies the workload trends seen in Section VI-A. For the compute-intensive WaterNSquared, the SSN achieves a in-order core speedup over the Mesh of 7%, but sees a 31% gain when O3 cores are utilized. For memory-intensive Radix and synchronization-sensitive Raytrace, the SSN realizes in-order speedups of 37% and 82%. Those gains further improve to 1.96× and 3.24× speedups with O3 cores. This shows that a SSN can be of even greater benefit if constructed for systems with O3 cores.

VII. 3-D Swizzle Switch Network

Three-dimensional integrated circuits (3-D ICs) are attractive options for overcoming the barriers in interconnect scaling. In a 3-D chip, multiple device layers are stacked together with direct vertical interconnects tunneling through them. The most important benefits of a 3-D chip over a traditional 2-D design is the reduction of global interconnects. We propose a combination of 3-D integration and the *Swizzle-Switch Network* to further improve performance.

A. 3-D Integration Technology

There are many techniques for 3-D integration. Simple techniques such as face-to-face bonding allow stacks of 2 chips while more complicated techniques that use wafer thinning allow for larger stack systems. For our analysis we use figures from a 3-D integration technology by Tezzaron [20]. Their technique is a via-first, back-end-of-line integration technology that has been demonstrated in several test chips [15], [28]. In the system of Fick *et al.* [15] the Tezzaron technology is used to stack four layers of CMOS logic on top of three layers of DRAM. The layers are thinned to less than 12 μm, an important

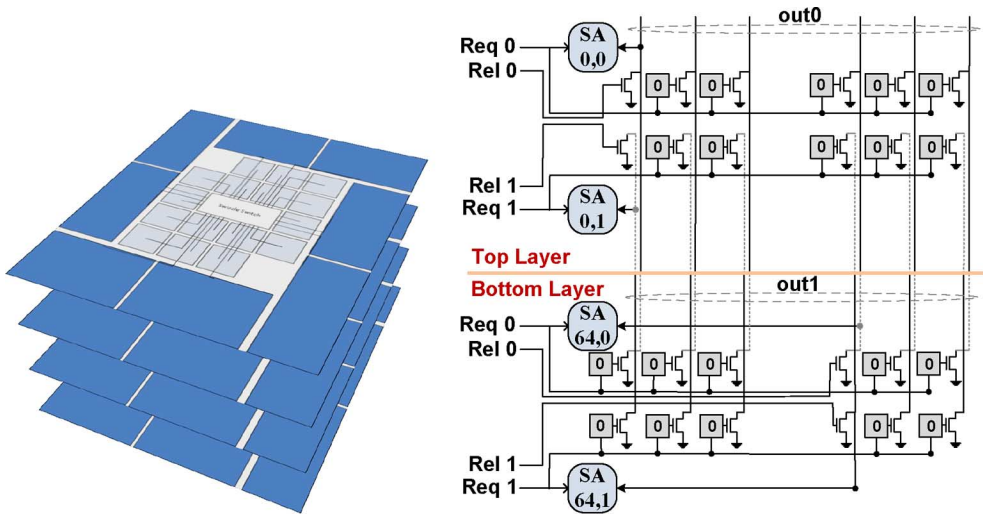


Fig. 18. (a) 3-D *Swizzle-Switch Network* achieved by stacking four 2-D *Swizzle-Switch Network* layers and using TSV's to interconnect the layers. (b) The modified circuits for the 3-D *Swizzle-Switch Network*. Even numbered outputs are arbitrated on the top layer, odd outputs are arbitrated on the bottom layer. Input request lines must be forwarded from the top $\rightarrow$ bottom or bottom $\rightarrow$ top through TSV connections. The total number of TSVs required is equal to the number of arbitrating crosspoints times the number of layers.

characteristic for reducing thermal resistance and *RC* delays. Additionally, the TSVs themselves are less than 6 μm thick while the size of each TSV is only 1.4 μm^2. As such, they can be placed with a density of 62 000 TSVs per square mm. The resistance ($< 0.35\ \Omega$) and capacitance (2 fF) of these TSVs are extremely small compared to other 3-D technologies. This allows for fast and short connections between layers. In fact, in a four-layer stack the length of a TSV running the whole height of the stack is $<$50 μm.

B. 3-D-SSN Architecture

We now turn to a 3-D-SSN design, where the basic 2-D-SSN is folded over multiple layers while holding the overall design constant at 64 cores. By folding the *Swizzle-Switch* over multiple layers the total size of the *Swizzle-Switch* is reduced and the shorter wires result in lower capacitance. This reduction in capacitance is translated into a speedup of the *Swizzle-Switch* itself. In addition the number of inputs and outputs per layer is reduced, leading to a more compact design where the links to and from the *Swizzle-Switch* are shorter and faster. The basic design we explore is presented in Fig. 18(a). In the four-layer system each layer contains 16 cores and 8 L2 banks. The central *Swizzle-Switch* communicates through TSVs to the other layers in the system. The modified circuit diagram is presented in Fig. 18(b). For illustrative purposes, we present only two inputs, two outputs, and two layers. The system requires that inputs on one layer forward requests to the proper output layer via TSVs. The total number of TSVs at each layer is equal to the total number of arbitrating crosspoints times the number of layers. The exploration of more sophisticated 3-D designs is left for future work.

C. Floorplanning

If the total area of the 2-D-SSN is split evenly across the 3-D layers it will result in a 50% and 75% reduction in area for a 2 and 4 layer system, respectively. However, the 2-D-SSN is already very dense in both wiring and logic. Currently the 2-D *Swizzle-Switch Network* system requires one additional unused routing lane per four lanes of dedicated routing to fit the required logic. The addition of TSVs in the system will dilate the size further. The total number of TSV's required per layer is equal to the total number of arbitrating crosspoints times the number of layers. Given the minimum spacing in the Tezzaron process, this corresponds to an additional routing track every eight tracks in the two-layer system and every four tracks in the four-layer system. This means the *Swizzle-Switch* in the two-layer 3-D-SSN is 57% of the size of a *Swizzle-Switch* in 2-D and the *Swizzle-Switch* in the four-layer 3-D-SSN is 32% of the size of a *Swizzle-Switch* in 2-D.

The reduced area of the SSN yields a faster design, and the smaller number of devices per layer shortens the links to the SSN. After careful floorplanning of the system, the two-layer version achieves an interconnect speed of 2.2 GHz, and the four-layer version achieves an interconnect speed of 2.7 GHz (core speeds remain at 1.5 GHz).

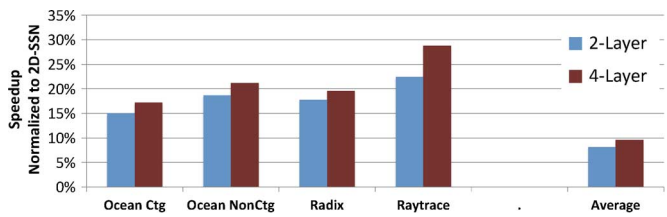


Fig. 19. Speedup of the 3-D-SSN on two-layer and four-layer systems compared to a 2-D-SSN. The benchmarks most sensitive to interconnect delay are plotted as well as the average across all benchmarks.

D. Performance Results

Fig. 19 presents the overall speedup of the 3-D-SSN normalized to the 2-D version. On average the 3-D system sees a 8% and 10% speedup for a two-layer and four-layer system, respectively. For benchmarks where the interconnect latency was critical—*Ocean*, *Radix*, and *Raytrace*—the improvements

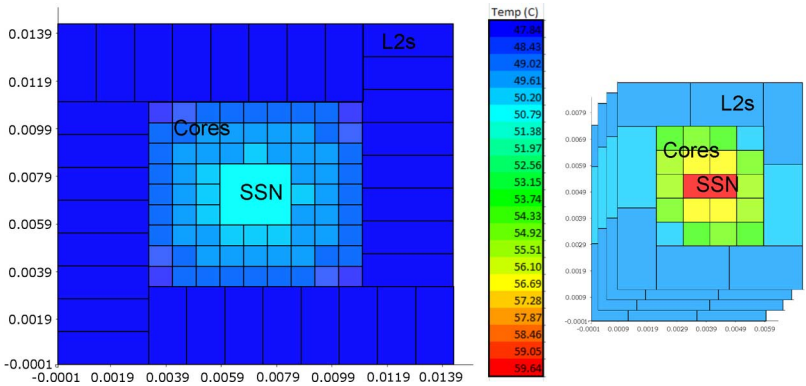


Fig. 20. HotSpot simulation of a 64 core SSN system on 1 and 4 layers for the worst case benchmark. The 1-layer system corresponds to a 2-D-SSN with a peak temperature of 51 ° C. The peak temperature of the 3-D chip is 60 ° C (The hottest layer is shown as the top of the 3-D stack).

are more significant showing a 15%–30% speedup. Recall from Section VI that these benchmarks improved significantly due to either faster L2 accesses, or faster synchronization and as such reflect further gains in a 3-D design. The other benchmarks only showed marginal sensitivity to interconnect latency and result in only modest gains 1%–6%, lowering the average across all benchmarks.

E. Thermal Analysis

As with any 3-D chip design, thermal constraints can be a matter of concern. To verify the system operates in a thermal region that can be cooled by conventional solutions, we perform analysis of the system with the HotSpot 5.1 [46] simulator. The thermal characteristics of the Tezzaron process were modeled in HotSpot and power draw numbers from the *Ocean* benchmark are plotted—the hottest benchmark. Power numbers for the Cortex-A5 were based on published data and scaled to 32 nm. Fig. 20 shows the simulated system for a single layer and a four-layer stack. The low power design of the Cortex-A5 processor helps to make stacking feasible. The peak temperature of the 1 and 4 layer systems is around 51 °C and 60 ° C, respectively, within the capability of passive cooling solutions. The HotSpot analysis did not consider the thermal dissipating characteristics of the TSVs, which would have further reduced the peak temperature.

VIII. Related Work

The paradigm shift toward many-core systems has led to a renewed interest in interconnect research and a transition from traditional bus-based systems [32] to more sophisticated topologies, including hierarchical bus models [7], [14], rings [19], [45], spidergon networks [8], mesh network-on-chips [4], [49], flattened butterfly on-chip networks [29], express cube on-chip networks [17], and crossbars [2], [26], [52]. The ability to provide uniform cache access latency makes crossbars an appealing option because predictable cache access latencies allow for quality-of-service guarantees and ease of programming. In addition, previous research has shown that crossbars can enable performance benefits in coherence protocols [11] as well as the construction of cache hierarchies [34]. While some studies have noted that link latency can increase to a point that it would be intolerable compared to an NoC system [9], [47], we prove through detailed floorplans and spice analysis that this is not the case.

The *Swizzle-Switch Network*(SSN) proposed in this paper optimizes the crossbar interconnect allowing for high performance many-core systems with minimal power and area overheads readily scaling to support 64-core systems. This work demonstrates the benefits of a crossbar-based architecture for systems that are required to support a wide range of communication patterns. Related works have also leveraged the benefits of high-radix switches by using narrow channels to increase crossbar radix and build larger systems from these high-radix building blocks [30], [44]. There has also been similar work analyzing crossbar interconnects for large-scale CMPs. Some assume idealized crossbars with minimal latencies to calculate best-case performance [40], others use crossbars to connect small clusters of cores in a hierarchical system [48], and yet others study pipelined/buffered crossbar systems [31], [37]. The SSN differs from these systems because it uses a flat, nonbuffered interconnect based on detailed floor-planning, SPICE simulation, and measured silicon results.

IX. Conclusion

As the number of on-chip cores increases, the demand for interconnection networks to simultaneously support high bisection bandwidth, predictable memory access latencies, and quality of service guarantees grows. NoC systems can provide high bandwidth but typically are unable to support applications requiring uniform memory access or quality of service due to high hop counts associated with these networks. Busses and crossbars can enable more consistent memory latencies but have traditionally been eschewed due to power and area concerns.

In this paper, we leverage a recently proposed crossbar building block, the *Swizzle-Switch*, to build high-radix interconnects, including a Flattened Butterfly and the *Swizzle-Switch Network*. These high-radix networks decrease the average and maximum hop counts in 64-core systems.

Using *Swizzle-Switches*, we have demonstrated that both FBFly and SSN topologies outperform Mesh networks. On average, the FBFly is 15% faster, has a 1.76× smaller L1 on-chip average miss latency, a 2.5× reduction in miss latency standard deviation, and a 10% energy savings over the Mesh.

The SSN improves system performance by 21%, with a 2.2× smaller L1 on-chip average miss latency, and a 3.0× reduction in miss latency standard deviation all while providing a 25% energy savings compared to the Mesh. These improvements show that the FBFly and SSN facilitate easier programmability and quality of service guarantees on many-core systems.

Finally, this paper demonstrates a 3-D integrated version of the SSN which increased the interconnect speed by 1.8× and improved overall runtime by up to 30% for latency sensitive benchmarks. Detailed thermal analysis also confirmed the energy-efficient design of the 3-D-SSN was able to operate within the range of passive cooling solutions.

REFERENCES

[1] A. Alameldeen and D. Wood, "Variability in architectural simulations of multi-threaded workloads," in *Proc. 9th Int. Symp. High-Performance Comput. Archit.*, 2003, pp. 7–18.

[2] J. Andrews and N. Baker, "Xbox 360 system arch," *IEEE Micro*, vol. 26, no. 2, pp. 25–37, Mar./Apr. 2006.

[3] ARM Ltd, Cortex-A5 Processor, ARM Databrief 2010.

[4] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proc. 20th Annu. Int. Conf. Supercomput.,*, 2006, pp. 187–198.

[5] N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of splash-2 and parsec," in *IEEE Int. Symp. Workload Characterizat.*, 2009, pp. 86–97.

[6] N. Binkert and B. Beckmann *et al.*, "The gem5 simulator," in *Comput. Architecture News*, Jun. 2011.

[7] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," in *ACM Comput. Surv.*, 2006.

[8] L. Bononi and N. Concer, "Simulation and analysis of network-on-chip architectures: Ring, spidergon and 2-D mesh," in *Proc. Design, Automat. Test Eur.*, 2006, vol. 2, p. 6.

[9] L. Bononi and N. Concer *et al.*, "Noc topologies exploration based on mapping and simulation models," in *Proc. 10th Euromicro Conf. Digital Syst. Design Archit., Methods Tools*, 2007, pp. 543–543.

[10] S. Borkar, "Networks for multi-core chips: A contrarian view," in *Int. Symp. Low Power Electron. Design*, 2007.

[11] L. Cheng and N. Muralimanohar *et al.*, "Interconnect-aware coherence protocols for chip multiprocessors," in *Proc. 33rd Int. Symp. Comput. Archit.*, 2006, pp. 339–351.

[12] D. Culler, J. Singh, and A. Gupta, *Parallel Computer Architecture: A Hardware/Software Approach*, 1st ed. San Francisco, CA: Morgan Kaufmann, 1998.

[13] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2003.

[14] R. Das and S. Eachempati *et al.*, "Design and evaluation of a hierarchical on-chip interconnect for next-generation cmps," in *Proc. IEEE 15th Int. Symp. High Performance Comput. Architect.*, Feb. 2009, pp. 175–186.

[15] D. Fick and R. Dreslinski *et al.*, "Centip3de: A 3930 dmips/w configurable near-threshold 3-D-stacked system w/64 arm cortex-m3 cores," in *IEEE Int. Solid- State Circuits Conf. Dige. Tech. Papers*, Feb. 2012, pp. 190–192.

[16] M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, vol. 17, no. 1, pp. 34–39, Jan. 1997.

[17] B. Grot and J. Hestness *et al.*, "Express cube topologies for on-chip interconnects," in *Proc. IEEE 15th Int. Symp. High Performance Comput. Archit.*, 2009, pp. 163–174.

[18] B. Grot, S. W. Keckler, and O. Mutlu, "Preemptive virtual clock: A flexible, efficient, and cost-effective QOS scheme for networks-on-chip," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit.*, New York, 2009, pp. 268–279.

[19] M. Gschwind and H. Hofstee *et al.*, "Synergistic processing in cell's multicore architecture," *IEEE Micro*, vol. 26, no. 2, pp. 10–24, Mar./Apr. 2006.

[20] S. Gupta and M. Hibert *et al.*, "Techniques for producing 3-D ics with high-density interconnect," in *21st Int. VLSI Multilevel Interconnect. Conf.*, 2004.

[21] R. Haring, "The IBM blue gene/q compute chip+simd floating-point unit," in *HotChips 23: Symp. High-Performance Chips*, 2011.

[22] J. Howard and S. Dighe *et al.*, "A 48-core ia-32 message-passing processor with DVFs in 45 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan. 2011.

[23] J. Hu and R. Marculescu, "Energy-and performance-aware mapping for regular NOC architectures," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, Apr. 2005.

[24] Intel Corp, Intel atom processor for nettop platforms, Intel product brief Nov. 2008.

[25] N. E. Jerger, L.-S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Proc. 35th Int. Symp. Computer Archit.*, 2008, pp. 229–240.

[26] T. Johnson and U. Nawathe, "An 8-core, 64-thread, 64-bit power efficient SPARC SOC (niagara2)," in *IEEE Int. Dig. Tech. Papers*, Feb. 2007, pp. 108–590.

[27] C. Kim, D. Burger, and S. W. Keckler, "An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches," in *Proc. 10th Int. Conf. Architectural Support Programm. Languages Operat. Syst.,*, 2002, pp. 211–222.

[28] D. H. Kim and K. Athikulwongse *et al.*, "3-D-maps: 3-D massively parallel processor with stacked memory," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2012, pp. 188–190.

[29] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2007, pp. 172–182.

[30] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a high-radix router," in *Proc. 32nd Int. Symp. Comput. Archit.*, Jun. 2005, pp. 420–431.

[31] G. Kornaros, "Bcb: A buffered crossbar switch fabric utilizing shared memory," in *Proc. 9th EUROMICRO Conf. Dig. Syst. Design: Archit., Methods Tools*, 2006, pp. 180–188.

[32] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling," in *Proc. 32nd Int. Symp. Comput. Archit.*, Jun. 2005, pp. 408–419.

[33] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *Proc. 35th Int. Symp. Comput. Archit.*, Jun. 2008, pp. 89–100.

[34] L. Li *et al.*, "Ccc: Crossbar connected caches for reducing energy consumption of on-chip multiprocessors," in *Proc. Euromicro Symp. Digital Syst. Design*, Sep. 2003, pp. 41–48.

[35] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *Proc. 34th Annu. Int. Symp. Comput. Archit.*, 2007, pp. 46–56.

[36] N. McKeown, "The ISLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Network.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.

[37] G. Passas, M. Katevenis, and D. Pnevmatikatos, "A $128 \times 128 \times 24$ gb/s crossbar interconnecting 128 tiles in a single hop and occupying 6% of their area," in *Proc. 4th ACM/IEEE Int. Symp. Networks-on-Chip*, May 2010, pp. 87–95.

[38] G. Passas, M. Katevenis, and D. Pnevmatikatos, "VLSI micro-architectures for high-radix crossbar schedulers," in *Proc. 5th IEEE/ACM Int. Symp. Networks Chip*, May 2011, pp. 217–224.

[39] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. 7th Int. Symp. High-Performance Comput. Archit.*, 2001, pp. 255–266.

[40] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," in *ACM Trans. Archit. Code Optim.*, May 2010, pp. 4:1–4:28.

[41] S. Satpathy and R. Dreslinski *et al.*, "Swift: A 2.1 tb/s 32×32 self-arbitrating manycore interconnect fabric," in *Proc. 2011 Symp. VLSI Circuits*, Jun. 2011, pp. 138–139.

[42] S. Satpathy and Z. Foo *et al.*, "A 1.07 tbit/s 128×128 swizzle network for SIMD processors," in *IEEE Symp. VLSI Circuits*, Jun. 2010, pp. 81–82.

[43] S. Satpathy and K. Sewell *et al.*, "A 4.5 tb/s 3.4 tb/s/w 64×64 switch fabric with self-updating least recently granted priority and quality of service arbitration in 45 nm CMOS," in *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers.*, Feb. 2012, pp. 478–480.

[44] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The blackwidow high-radix CLOS network," in *Proc. 33rd Int. Symp. Comput. Archit.*, 2006, pp. 16–28.

[45] L. Seiler and D. E. A. Carmean, "Larrabee: A many-core ×86 architecture for visual computing," in *ACM Trans. Graphics*, 2008.

[46] K. Skadron and M. Stan *et al.*, "Temperature-aware microarchitecture," in *Proc. 30th Annu. Int. Symp. Comput. Archit.*, Jun. 2003, pp. 2–13.

[47] J. C. Villanueva *et al.*, "A performance evaluation of 2-D-mesh, ring, and crossbar interconnects for chip multi-processors," in *Proc. 2nd Int. Workshop Network Chip Archit.*, Dec. 2009, pp. 51–56.
[48] H.-S. Wang, L.-S. Peh, and S. Malik, "A power model for routers: Modeling alpha 21364 and infiniband routers," *IEEE Micro*, vol. 23, no. 1, pp. 26–35, Jan./Feb. 2003.
[49] D. Wentzlaff and P. Griffin *et al.*, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep.–Oct. 2007.
[50] S. C. Woo and M. Ohara *et al.*, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. 22nd Annu. Int. Symp. Comput. Archit.*, Jun. 1995, pp. 24–36.
[51] Y. Zhang and M. J. Irwin, "Power and performance comparison of crossbars and buses as on-chip interconnect structures," in *Conf. Rec. 33rd Asilomar Conf. In Signals, Syst., Comput.*, Oct. 1999, vol. 1, pp. 378–383.
[52] Y. P. Zhang *et al.*, "A study of the on-chip interconnection network for the IBM cyclops64 multi-core architecture," in *Proc. 20th Int. Parallel Distrib. Process. Symp.*, Apr. 2006, p. 10.

Korey Sewell received the B.S. degree in computer science from the University of California, Riverside, in 2004, and the M.S.E. degree, in 2007, in computer science and engineering from the University of Michigan, Ann Arbor, where he is currently a doctoral candidate.

He has research interests in high-performance microprocessor design, many-core systems, on-chip interconnects, and simulation modeling.

Ronald G. Dreslinski received the B.S.E. degree in electrical engineering, the B.S.E. degree in computer engineering, and the M.S.E. and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor.

He is currently a Research Scientist at the University of Michigan. His research focuses on architectures that enable emerging low-power circuit techniques.

Thomas Manville received the B.S. degree in computer engineering, in 2011, from the University of Michigan, Ann Arbor, where he is currently working toward the M.S. degree under Prof. T. Mudge.

His current research interests include memory controllers, branch prediction and scalable, on chip interconnects.

Sudhir Satpathy received the B.S. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 2007, and the Ph.D. degree in electrical engineering from the University of Michigan, Ann Arbor, in 2011.

Currently, he serves as a Research Scientist at Intel's Circuits Research Lab, Hillsboro, OR. His primary research interests are on-die interconnect fabrics, arithmetic and DSP circuit design. He has published 10 conference papers, won two student design contest awards, and holds one issued and five pending U.S. patents.

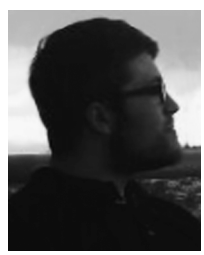
Nathaniel Pinckney received the B.S. degree from Harvey Mudd College, Claremont, CA, in 2008. He is a graduate student at the University of Michigan, Ann Arbor.

He worked two years at Sun Microsystems' Sun Labs. His interests include low-power near-threshold circuits.

Geoff Blake received the B.S.E., M.S.E., and Ph.D. degrees in computer science and engineering from the University of Michigan, Ann Arbor, in 2004, 2006, and 2011, respectively.

His research interests include multi-core architecture, operating systems, transactional memory, and more recently enterprise applications for low power servers. He is currently working in research and development at ARM Inc.

Dr. Blake is a member of the ACM.



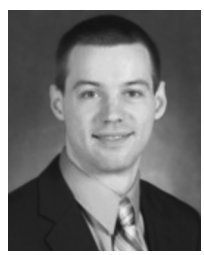
Michael Cieslak received the B.S. degree in computer engineering from the University of Michigan, Ann Arbor, in 2011.

He has conducted research in the field of scalable interconnect design under Prof. T. Mudge. He currently works at Amazon, where his focus is on improving the third party seller's experience.

Reetuparna Das received the Ph.D. degree in computer science and engineering from the Pennsylvania State University, University Park.

She is a research faculty in the Computer Science and Engineering Department at the University of Michigan, Ann Arbor, and a member of the Advanced Computer Architecture Lab (ACAL). Prior to this, she was a Research Scientist at Intel Labs, Santa Clara, CA. Her research interests include computer architecture and its interaction with software systems and device/VLSI technologies.

Dr. Das received an IEEE Top Picks award, outstanding research and teaching assistantship awards from the Computer Science and Engineering Department at Pennsylvania State University.



Thomas F. Wenisch (M'07) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA.

He is the Morris Wellman Faculty Development Assistant Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. His research interests include computer architecture, server and data center energy efficiency, smartphone architecture, and multiprocessor systems.

Dr. Wenisch is a member of ACM.

Dennis Sylvester (S'95–M'00–SM'04–F'11) received the Ph.D. degree in electrical engineering from the University of California, Berkeley, where his dissertation was recognized with the David J. Sakrison Memorial Prize as the most outstanding research in the UC-Berkeley Electrical Engineering and Computer Science Department.

He is a Professor of Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor, and Director of the Michigan Integrated Circuits Laboratory (MICL), a group of ten faculty and 60+ graduate students. He previously held research staff positions in the Advanced Technology Group of Synopsys, Mountain View, CA, Hewlett-Packard Laboratories, Palo Alto, CA, and a visiting professorship in Electrical and Computer Engineering at the National University of Singapore. He has published over 300 articles along with one book and several book chapters. His research interests include the design of millimeter-scale computing systems and energy efficient near-threshold computing for a range of applications. He holds 10 U.S. patents. He also serves as a consultant and technical advisory board member for electronic design automation and semiconductor firms in these areas. He co-founded Ambiq Micro, a fabless semiconductor company developing ultra-low power mixed-signal solutions for compact wireless devices.

David Blaauw (F'12) received the B.S. degree in physics and computer science from Duke University, Durham, NC, in 1986, and the Ph.D. degree in computer science from the University of Illinois, Urbana, in 1991.

Until August 2001, he worked for Motorola, Inc., Austin, TX, were he was the manager of the High Performance Design Technology Group. Since August 2001, he has been on the faculty at the University of Michigan where he is a Professor. He has published over 350 papers and hold 40 patents. His work has focused on VLSI design with particular emphasis on ultra low power and high performance design.

Dr. Duke was the Technical Program Chair and General Chair for the International Symposium on Low Power Electronic and Design. He was also the Technical Program Co-Chair of the ACM/IEEE Design Automation Conference and a member of the ISSCC Technical Program Committee.

Trevor Mudge (S'74–M'77–SM'84–F'95) received the B.Sc. degree from the University of Reading, Reading, U.K., in 1969, and the M.S. and Ph.D. degrees in computer science from the University of Illinois, Urbana, in 1973 and 1977, respectively.

Since 1977, he has been on the faculty of the University of Michigan, Ann Arbor. He was named the first Bredt Family Professor of Electrical Engineering and Computer Science after concluding a 10 year term as the Director of the Advanced Computer Architecture Laboratory—a group of eight faculty and about 60 graduate students. He is author of numerous papers on computer architecture, programming languages, VLSI design, and computer vision. He has also chaired about 50 theses in these areas. His research interests include computer architecture, computer-aided design, and compilers. In addition to his position as a faculty member, he runs Idiot Savants, a chip design consultancy.

Dr. Mudge is member of the ACM, the IET, and the British Computer Society.