SWIFT: A Low-Power Network-On-Chip Implementing the Token Flow Control Router Architecture With Swing-Reduced Interconnects

Jacob Postman, *Student Member, IEEE*, Tushar Krishna, *Student Member, IEEE*, Christopher Edmonds, Li-Shiuan Peh, *Member, IEEE*, and Patrick Chiang, *Member, IEEE*

***Abstract*— A 64-bit, 8 × 8 mesh network-on-chip (NoC) is presented that uses both new architectural and circuit design techniques to improve on-chip network energy-efficiency, latency, and throughput. First, we propose token flow control, which enables bypassing of flit buffering in routers, thereby reducing buffer size and their power consumption. We also incorporate reduced-swing signaling in on-chip links and crossbars to minimize datapath interconnect energy. The 64-node NoC is experimentally validated with a 2 × 2 test chip in 90 nm, 1.2 V CMOS that incorporates traffic generators to emulate the traffic of the full network. Compared with a fully synthesized baseline 8 × 8 NoC architecture designed to meet the same peak throughput, the fabricated prototype reduces network latency by 20% under uniform random traffic, when both networks are run at their maximum operating frequencies. When operated at the same frequencies, the SWIFT NoC reduces network power by 38% and 25% at saturation and low loads, respectively.**

***Index Terms*— Architecture, circuits, interconnect, low-power design, on-chip networks, routing.**

I. Introduction

NETWORKS-ON-CHIP (NoC) form the on-chip communication backbone between processing cores in many core processor architectures. Aided by Moore's law, increasing core counts have become the answer to meet high performance demands at low-power budgets, rather than designing increasingly complex cores. However, for the tens to hundreds of on-chip cores to become a reality, the NoC needs to be highly scalable, and provide low-latency, high-throughput communication in an energy-efficient manner. Multicore research prototypes, such as MIT's 16-core RAW [1], UT Austin's 40-core TRIPS [2], and Intel's 80-core TeraFLOPS [3] have adopted tile-based homogenous NoC architectures that simplify the interconnect design and verification. The network routers are laid out as a mesh, and communicate with one another in a packet switched manner. While these designs demonstrated the potential of multicore architectures, they also exposed the rising contribution of the network toward the total power consumed by the chip. For instance, 39% of the 2.3-W power budget of a tile is consumed by the network itself in Intel's TeraFLOPS. If not addressed, this trend will become the major stumbling block for the further scaling of processor counts. This paper addresses some of these concerns. NoC routers primarily consist of buffers, arbiters, and a crossbar switch, and are interconnected with one another via links. Buffers are required to prevent collisions between packets that share the same links, and are managed by the control path. The crossbar switches and the links form the datapath through which actual data transmission occurs. Prior NoC prototypes have observed that the primary contributers to NoC power are the buffers (31% in RAW [1], 35% in TRIPS [2], 22% in TeraFLOPS [3]), which are typically built out of SRAM or register files; the crossbar switches (30% in RAW [1], 33% in TRIPS [2], 15% in TeraFLOPS [3]); and the core-core links (39% in RAW [1], 31% in TRIPS [2], 17% in TeraFLOPS [3]).

Significant research in NoC router microarchitectures has focused on performance enhancement (low-latency and high-throughput). The MIT RAW [1], and its subsequent commercial version, the Tilera TILEPro64 [4] use multiple physical networks to improve throughput, each interconnected via very simple low-latency routers, presenting one-cycle delay in the router to traffic going straight and two-cycles for turning traffic. The UT TRIPS [2] and Intel TeraFLOPS [3] enhance throughput using virtual channels (VC)[1] [5]. However, VCs increase the arbitration complexity because: 1) there are more contenders for the input and output ports of the switch within each router and 2) VC arbitration is required in addition to switch arbitration. This forces all flits[2] to get buffered, perform successful arbitrations, and then proceed through the switch and the links to the next router, increasing both the delay and the dynamic power consumption within each router.

Manuscript received October 12, 2011; revised May 17, 2012; accepted July 21, 2012. Date of publication August 31, 2012; date of current version July 22, 2013. This work was supported in part by the National Science Foundation under Grant CCF-0811375 and Grant CCF-0811820, an NSF Doctoral Fellowship, a DoE Early CAREER Award, the MARCO Interconnect Focus Center, the Gigascale Systems Research Center, and gift donations from Intel Corporation.

J. Postman and P. Chiang are with the Department of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97333 USA (e-mail: postmaja@eecs.oregonstate.edu; pchiang@eecs.oregonstate.edu).

T. Krishna and L.-S. Peh are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: tushar@csail.mit.edu; peh@csail.mit.edu).

C. Edmonds is with Microsoft Corporation, Redmond, WA 98052-6399 USA (e-mail: edmondsc@gmail.com).

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/TVLSI.2012.2211904

[1] VCs are analogous to turning lanes on highways to avoid traffic leaving different output ports from blocking each other, known as head-of-line blocking.

[2] Flits are small fixed-sized sub-units within a packet, equal to the router-to-router link width.

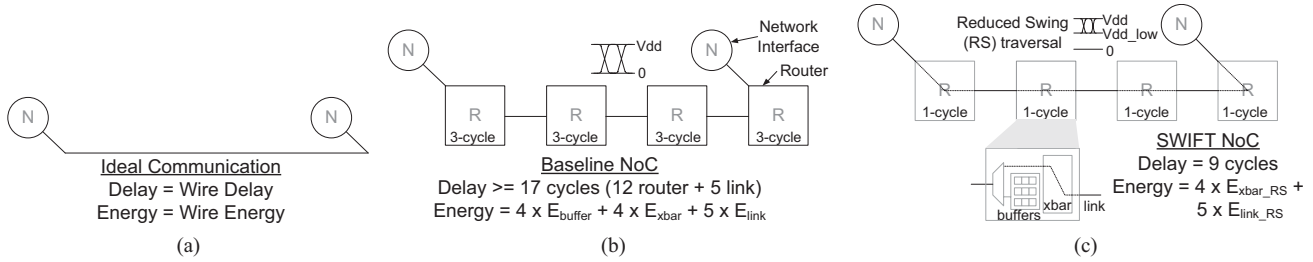


Fig. 1. Overview of SWIFT NoC versus a baseline NoC and ideal communication. (a) Ideal communication. (b) Baseline NoC. (c) SWIFT NoC.

UT TRIPS [2] uses a three-cycle router pipeline, while Intel TeraFLOPS [3] uses a five-cycle router pipeline, followed by an additional cycle in the link. Even though these routers could potentially be designed to run slower and finish all tasks within a cycle, flits would still have to be buffered (and perhaps read out in the same cycle upon successful arbitration).

To mitigate the long latencies in these multicycle VC routers, recent research in NoCs has suggested ways to perform speculative arbitration [6], [7] or nonspeculative arbitration by sending advanced signals [8]–[11], before the flit arrives, to allow flits to traverse the switch directly, instead of getting buffered. This presents a single-cycle router delay to most flits, and reduces the power consumed in buffer writes/reads. However, none of these designs have been proven in silicon yet.

In this paper, we present the SWing-reduced Interconnect For a Token-based NoC (SWIFT NoC), which is a novel architecture/circuit co-design that targets power reduction in both the control path (buffers) and the datapath (crossbar and links). On the architecture front, this is the first fabricated NoC prototype to target an aggressive one-cycle VC router, with intelligent nonspeculative network flow-control (token flow control [8]) to bypass router buffers. On the circuit front, we identify that datapath traversal is imminent in any router (speculative/nonspeculative single or multicycle) and custom design a crossbar with low-swing links, a first of its kind, and interface it with low-swing router-to-router links, to provide a reduced-swing datapath. In contrast, previous implementations of low-swing on-chip wires have restricted their use to global signaling [12], [13]. Together, the buffer bypassing flow control, and the reduced-swing crossbar and links, provide two-cycle-per-hop on-chip communication from the source to the destination over a reduced-swing datapath. Compared to a state-of-the-art baseline NoC (which we model similar to UT TRIPS [2] and TeraFLOPS [3]) operating at its peak frequency and designed to meet the same saturation throughput, the SWIFT NoC lowers low-load latency by 20% with uniform random traffic. When running both the baseline and SWIFT at the same frequency, the low-load latency savings are 39%. In addition, SWIFT reduces peak control path power by 49%, and peak datapath transmission power by 62%, giving a total power reduction of 38% compared to the baseline running at the same throughput. A high-level overview of the SWIFT NoC is shown in Fig. 1.

To experimentally validate the entire NoC co-designed with the proposed microarchitecture and circuit techniques, the SWIFT router is designed in Verilog and implemented using a commercial 90 nm, 1.2 V CMOS standard cell library. Next, the synthesized router logic is manually integrated with the custom, reduced-swing crossbars and links. While the proposed NoC is designed as an 8×8 2-D mesh, the fabricated test-chip is comprised of a smaller, 2×2 mesh subsection of four routers, verifying the design practicality, and validating the network simulation results. At 225 MHz, the total power savings of the entire network were measured to be 38% versus a baseline, fully-synthesized, network design operating at the same peak throughput.

The rest of this paper is organized as follows. Section II provides the motivation for this paper. Section III presents the microarchitecture of the SWIFT router. Section IV describes the reduced-swing crossbar and link circuits. Section V presents and analyzes our results. Section VI presents some of our learnings and insights. Section VII concludes.

II. Motivation

The ideal energy-delay of a NoC traversal should be that of just the on-chip links from the source to the destination, as shown in Fig. 1(a). However, this would require each core to have a direct connection with all other cores which is not scalable. Packet-switched NoCs [5] share links by placing routers at the intersections to manage buffering, arbitration for the crossbar switch, and routing of the flits in order to avoid collisions while the link is occupied, at the cost of increased delay and power, as shown in Fig. 1(b). In this paper, we make two key observations.

First, fabricated NoC prototypes in the past have typically used relatively simple VC router architectures that do not optimize buffer utilization. Data are always buffered without regard to route availability. This dissipates unnecessary energy, and introduces additional latency within a router pipeline. Even if the datapath is available, all packets go through multiple pipeline stages within each router, including buffering, routing, switch allocation, and switch (crossbar) traversal; this is then followed by the link (interconnect) traversal. We believe that the router control path can be optimized to utilize the available link bandwidth more efficiently by exploiting adaptive routing and intelligent flow-control, thus enhancing throughput. This architectural enhancement allows many flits to bypass buffering, even at high traffic rates. By obviating the need for buffer reads/writes, fewer buffers are required to sustain a target bandwidth. The result is a reduction in both

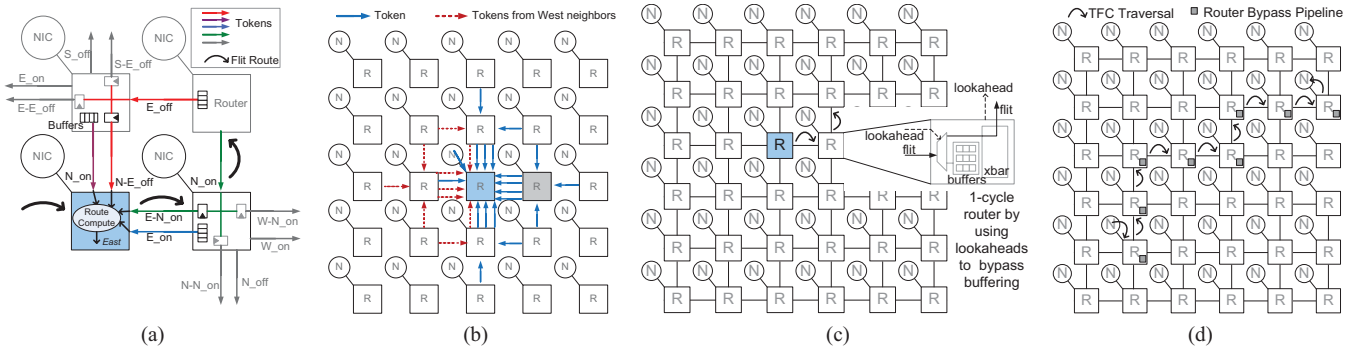


Fig. 2. SWIFT NoC architecture. (a) Routing using tokens. (b) Token distribution. (c) Bypass flow-control using lookaheads. (d) One-cycle router+ one-cycle LT.

the router latency and buffer read/write power. This approach has been explored in academic literature [6]–[10], but has not made it to main stream design.

Second, when buffering is bypassed, the energy consumed by a flit moving through the network will be dominated by both the switch and link traversal. Hence, implementing low-power crossbar and link circuits becomes even more critical for reducing total network power. There have been several recent papers that demonstrate low-swing global links in isolation [12]–[15], but only [15] has shown integration of low-swing links between cores within a practical SoC application. Previously reported crossbar designs have utilized partial activation [15], segmentation [16], and dynamic voltage scaling [17] to reduce crossbar power; however, this is the first work in which low-swing links are utilized within the densely-packed crossbar itself and has recently been extended by automating the generation of low-swing crossbar and link circuits for modular application to NoCs [18] that was outside the scope of this paper.

The SWIFT NoC applies both these observations.

1) We enhance a state-of-the-art router pipeline with the ability to form adaptive bypass paths in the network using tokens [4]. Flits on bypass paths use one-cycle pipelines within the router, while other flits use a three-stage pipeline. This reduces buffer read/write power in addition to lowering latency.
2) We custom design the router crossbar and core-to-core links with reduced-swing wires to lower data transmission energy.

III. SWIFT Architecture: Bypass Flow Control

NoC design primarily involves the following components: routing, flow-control, and router microarchitecture. Routing determines which links the packet traverses from source to destination. Flow-control determines when the packet can traverse its links. Router microarchitecture implements the routing and flow-control logic in a pipelined manner and houses the crossbar switch to provide connectivity across different directions. We describe the architectural design of SWIFT NoC in detail, which is based on token flow control (TFC) [8], providing relevant background where necessary.

A. Related Work on NoC Prototypes

The TILEPro64 [4] from Tilera (inspired by MITs RAW processor [1]) is a 64-core chip that uses five separate 8×8 mesh networks. One of these networks is used for transferring pre-defined static traffic, while the remaining four carry variable-length dynamic traffic, such as memory, I/O, and user-level messages. The TRIPS [2] chip from UT Austin uses two networks, a 5×5 operand network to replace operand bypass and L1 Cache buses, and a 4×4 on-chip network (OCN) to replace traditional memory buses. The Intel TeraFLOPS [3] 80-core research chip uses an 8×10 network for memory traffic. Table I compares the design components for these three prototypes for their multiflit memory networks. The table also summarizes the design of our state-of-the-art Baseline NoC (designed for comparison purposes similar to UT TRIPS [2] and Intel TeraFLOPS [3]), which will be described later in Section V-B. The three prototypes used text book routers [5] with simple flow control algorithms, as their primary focus was on demonstrating a multicore chip with a nonbus and nonring network. In the SWIFT NoC project, we take a step further, and explore a more optimized network design, TFC [8], with reduced-swing circuits in the datapath. We simultaneously address network latency (buffer bypassing, one-cycle router), throughput (adaptive routing, buffer bypassing at all traffic levels using tokens and lookaheads), and power (buffer bypassing, low-swing interconnect circuits, clock gating). The SWIFT NoC optimizations can potentially enhance the simple networks of all these multicore prototypes.

B. Routing With Tokens

In the SWIFT NoC, every input port sends a one-bit token to its neighbor, which is a hint about buffer availability at that port. If the number of free buffers is greater than a threshold (which is three in order to account for flits already in flight), the token is turned ON (by making the wire high), else it is turned OFF. The neighbor broadcasts this token further to its neighbors, along with its own tokens. Flits use these tokens to determine their routes. They try to adapt their routes based on token availability. Fig. 2(a) shows an example of this. The shaded router receives tokens from its N, E, and NE neighbors. The incoming flit chooses the East output port over the North output port based on token availability. We implement minimal

TABLE I
COMPARISON OF NoC DESIGNS

Characteristic	TILEPro64 [4] (1 sta, 4 dyn nets)	UT TRIPS [2] (OCN)	Intel TeraFLOPS [3]	Baseline*	SWIFT
Process parameters					
Technology	90 nm	130 nm	65 nm	90 nm	90 nm
Chip frequency	700–866 MHz	366 MHz	5 GHz	Not available	400 MHz
Router area	Not available	1.10 mm^2	0.34 mm^2	0.48† mm^2	0.48 mm^2
Network parameters					
Topology	8×8 mesh	4×10 mesh	8×10 mesh	8×8 mesh	8×8 mesh‡
Flit size	32 b	138 b	39 b	64 b	64 b
Message length	1–128 flits	1–5 flits	2 or higher flits	5 flits	5 flits
Routing	X-Y dimensional order	Y-X dimension order	Source	X-Y dimension order	Adaptive (West-first)
Flow control	Wormhole	Wormhole with VCs	Wormhole with VCs	Wormhole with VCs	TFC [8]
Buffer management	Credit-based	Credit-based	On/Off	On/Off	TFC [8]
Router parameters					
Ports	5	6	5	5	5
VCs per port	0 (5 separate networks)	4	2	2 and 4	2
Buffers per port	12 (3/dynamic net)	8	32	8 and 16	8
Crossbar	5×5	6×6	5×5	5×5	5×5

*Not fabricated, only laid out for comparison purposes.

†Baseline tile was given same area as SWIFT for place-and-route.

$^{\ddagger}2 \times 2$ mesh for test chip.

routing, with a west-first turn rule [5] to avoid deadlocks. Any other adaptive routing algorithm can be used as well.

Each token is forwarded upto three-hops, via registers at each intermediate router. Tokens are also forwarded up to the network interfaces (NIC) at each router. The number three was fixed based on experiments which can be found in the TFC paper [8]. Intuitively, deeper token neighborhoods do not help much since the information becomes stale with each hop. Moreover, the route is updated at every hop based on the tokens at that router, and the flit only needs to choose between a maximum of two output ports (for minimal routing). Adding more tokens would add more wires and registers without returning much benefit.

For illustration purposes, Fig. 2(b) shows the token distribution relative to the shaded router in a two-hop neighborhood. 16 tokens enter the shaded router from a two-hop neighborhood, plus one from the local port. However, West-first routing algorithm allows us to remove tokens from the west neighborhood (except the immediate neighbor) since a packet has to go west irrespective of token availability, reducing the total tokens from $(16 + 1)$ to $(11 + 1)$. Similarly, there are a total of $(36 + 1)$ tokens in a three-hop neighborhood. Removing the west tokens allows us to reduce this number to $(22 + 1)$ bits of tokens per router and these act as inputs to the combinational block that performs route computation.

C. Flow Control With Lookaheads

Conventional flow control mechanisms involve arbitration for the crossbar switch among the buffered flits. Some prior works [4], [7]–[10], [19] propose techniques to allow flits that have not yet arrived to try and pre-allocate the crossbar. This enables them to bypass the buffering stage and proceed directly to the switch upon arrival. This not only lowers traversal latency, but also reduces buffer read/write power. The SWIFT NoC implements such an approach, based on TFC [8], as shown in Fig. 2(c). TFC [8] has been shown to be better than other approaches like express virtual channels (EVC) [9] as it allows flits to chain together tokens to form arbitrarily long bypass paths with turns, while EVC only allowed bypassing within a dimension upto a maximum of three-hops. Other approaches to tackle buffer power include adding physical links to bypass intermediate routers [20], or using link repeaters as temporary buffers [4], [19], [21] to reduce buffers within the router. These techniques can enhance energy-delay further at the cost of more involved circuit design.

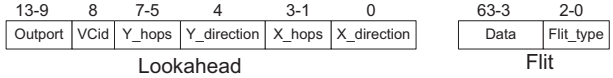


Fig. 3. Lookahead and flit payloads.

In the SWIFT NoC, the crossbar is pre-allocated with the help of lookahead signals, which are 14-bit signals sent for each flit, one-cycle before it reaches a router. The implementation of the lookahead generation and traversal to enable a one-cycle advanced arrival will be explained later in Section III-D.

A lookahead is prioritized over locally-buffered flits, such that a local switch allocation is killed if it conflicts with a lookahead. If two or more lookaheads from different input ports arrive and demand the same output port, a switch priority pointer at the output port (which statically prioritizes each input port for an epoch of 20 cycles for fairness) is used to decide the winner and the other lookaheads are killed. The flits

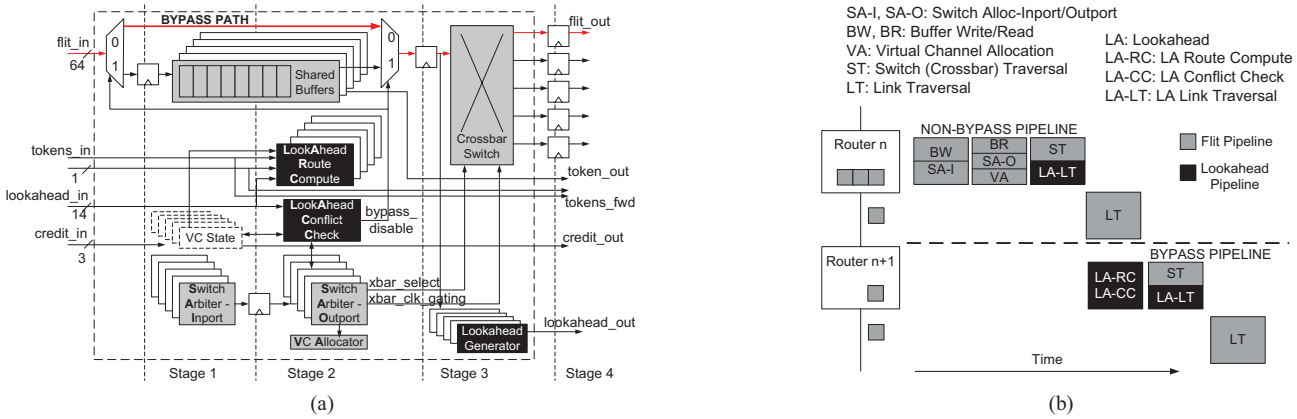


Fig. 4. SWIFT architectural design. (a) SWIFT router microarchitecture. (b) SWIFT nonbypass and bypass pipelines.

corresponding to the killed lookaheads get buffered similar to the conventional case. Since the bypass is not guaranteed, a flit can proceed only if the token from the neighbor is ON (indicating an available buffer).

The lookahead and flit payloads are shown in Fig. 3. Lookaheads carry information that would normally be carried by the header fields of each flit: destination coordinates, input VC id, and the output port the corresponding flit wants to go out from. They are thus not strictly an overhead. Lookaheads perform both switch allocation, and route computation.

The SWIFT flow control has three major advantages over previous prototypes with simple flow control.

1) *Lower Latency:* Bypassing obviates the buffer write, read, and arbitration cycles.
2) *Fewer Buffers:* The ability of flits to bypass at all loads keeps the links better utilized while minimizing buffer usage, and reducing buffer turnaround times. Thus, the same throughput can be realized with fewer buffers.
3) *Lower Power:* Requiring fewer buffers leads to savings in buffer power (dynamic and leakage) and area, while bypassing further saves dynamic switching energy due to a reduction in the number of buffer writes and reads.

The SWIFT NoC guarantees that flits within a packet do not get re-ordered. This is ensured by killing an incoming lookahead for a flit at an input port if another flit from the same packet is already buffered. Pt-to-pt ordering is however not guaranteed by SWIFT. This is because lookaheads are prioritized over locally buffered flits, which could result in two flits from the same source to the same destination getting re-ordered if the first one happened to get buffered at some router while the second one succeeded in bypassing that router. Most OCN designs use multiple virtual networks to avoid protocol level deadlocks. While request virtual networks often require pt-to-pt ordering for consistency reasons, response virtual networks often do not place this constraint, and TFC can be used within these virtual networks.

A potential network traversal in the SWIFT NoC using tokens and lookaheads is shown in Fig. 2(d).

D. Router Microarchitecture

SWIFT tries to present a one-cycle router to the data by performing critical control computations off the critical path.

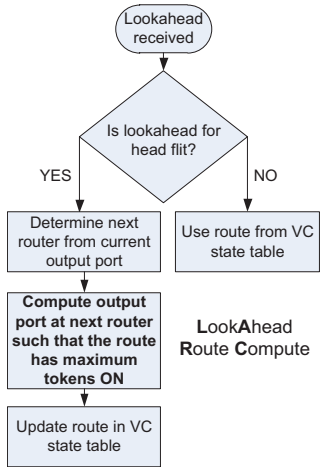


Fig. 5. Flow chart for LA-RC.

The modifications over a baseline router are highlighted in black in Fig. 4(a). In particular, each SWIFT router consists of two pipelines: a nonbypass pipeline which is three-stages long (and the same as a state-of-the-art baseline), and a bypass pipeline, which is only one-stage and consists of the crossbar traversal. The router pipeline is followed by a one-cycle link traversal. The three-stage baseline pipeline is described in the appendix.

Fig. 4(b) shows the pipeline followed by the lookaheads to enable them to arrive a cycle before the flit, and participate in the switch allocation at the next router. All flits try to use the bypass pipeline at all routers. The fallback is the baseline three-stage nonbypass pipeline.

1) Lookahead Route Compute (LA-RC): The lookahead of each head flit performs a route compute (LA-RC) to determine the output port at the next router [22]. This is an important component of bypassing because it ensures that all incoming flits at a router already know which output port to request, and whether to potentially proceed straight to ST. We use West-first routing, an adaptive-routing algorithm that is deadlock free [5]. The adaptive-routing unit is a combinational logic block that computes the output port based on the availability of the tokens from three-hop neighboring routers, rather than use local congestion metrics as indication of traffic. An overview of LA-RC is shown in Fig. 5.

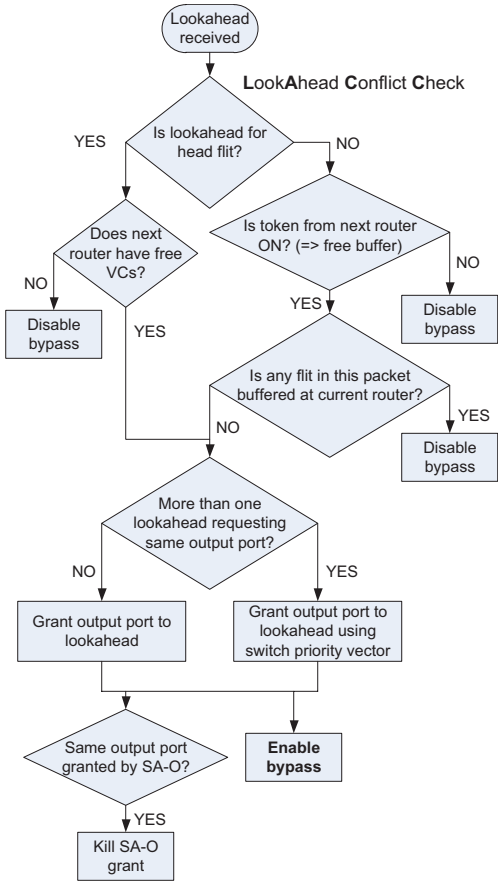


Fig. 6. Flow chart for LA-CC.

2) Lookahead Conflict Check (LA-CC): The lookahead places a request for the output port in the LA-CC stage, which grants it the output port unless there is a conflict or the output port does not have free VCs/buffers. An overview of LA-CC is shown in Fig. 6. LA-CC occurs in parallel to the SA-O stage of the nonbypass pipeline, as shown in Fig. 4(a). A lookahead is given preference over the winners of SA-O, and conflicts between multiple lookaheads are resolved using the switch priority vector described earlier in Section III-C. Muxes connect the input ports of the winning lookaheads directly to the crossbar ports. The corresponding flits that arrive in the next cycle bypass the buffers, as shown in Fig. 4(a). Any flits corresponding to killed lookaheads, meanwhile, get buffered and use the nonbypass pipeline.

3) Lookahead Link Traversal (LA-LT): While the flit performs its crossbar traversal, its lookahead is generated and sent to the next router. All the fields required by the lookahead, shown in Fig. 3, are ready by the end of the previous stage of LA-RC and LA-CC. Fig. 4(b) shows how the lookahead control pipeline stages interact with the flit pipeline stages in order to realize a one-cycle critical datapath within the router.

IV. SWIFT CIRCUITS: LOW-SWING ON-CHIP WIRES

When flits are able to bypass buffering, SWIFT's pipeline reduces to the one-cycle bypass pipeline comprised of just one-cycle switch traversal (ST), and one-cycle link traversal (LT) that results in two cycles per hop. These two stages correspond to the data movement that flits take through the crossbar switch and through the core-to-core interconnect, respectively. Crossbars provide the physical connection between input and output router ports, allowing the flow of data to be directed by routing logic. Core-to-core links provide the communication channel between adjacent network routers.

Unlike locally connected logic that drive relatively short, locally-routed wires, crossbars and links are primarily composed of tightly-packed, parallel wires that traverse longer distances with close inter-wire coupling. The energy consumed in these components is dominated by the dynamic switching of these wire capacitances rather than transistor gate input capacitances. When flit buffering is bypassed within the router, the energy required to drive these wire capacitances quickly begins to dominate the network power consumption. Low-voltage swing circuit techniques provide an energy-efficient alternative to full-swing digital CMOS signaling that reduces the dynamic energy of interconnect wires without sacrificing performance.

A. Related Work on On-Chip Signaling Techniques

Previous works, such as [17], have explored the use of conventional dynamic voltage scaling for reducing crossbar and link power in standard digital logic implementations. Though techniques based on standard logic cells are simpler to implement than custom circuits, they suffer from two major disadvantages. First, in order to reduce the supply voltage, the operating frequency must also be reduced. This limits the savings that can be achieved in signaling energy to times during which the network utilization is low enough that reduced performance can be tolerated. Second, the reduction in voltage signal swing on the interconnect wires is limited to the minimum supply voltage that all attached logic circuits will function at. Thus, custom circuits for low-voltage swing signaling are required in order to reduce wire energy while still maintaining performance.

Custom circuit techniques have targeted improved mW/Gb/s on multi-gigabit/second serial links for on-chip wires up to 10 mm. In these cases, signal swings as low as 120 mV are used on global differential RC wires [12], [13] or transmission line structures [23] either for transporting data directly or for sending control signals to efficiently allocate network buffers and links [24]. However, in order to achieve high performance across long distances, these designs require large transceiver areas of 887 μm^2 [12] and 1760 μm^2 [13]. These area overheads make them largely unsuitable for the shorter, wide data busses within NoC core-core links and crossbars.

Alternative energy-efficient, low-swing signaling techniques are needed for the shorter wires and dense transceiver requirements in NoC topologies. Capacitively-coupled transmitters are explored in [25], achieving speeds of 5–9 Gb/s across 2-mm links. However, the feed-forward capacitor increases transmitter area and requires DC-balanced coding or resistive DC biasing to prevent common-mode drift due to the intrinsic AC-coupling. Current-mode signaling, [14], dissipates constant static current throughout the entire clock period, mitigating its energy-efficiency figure-of-merit.

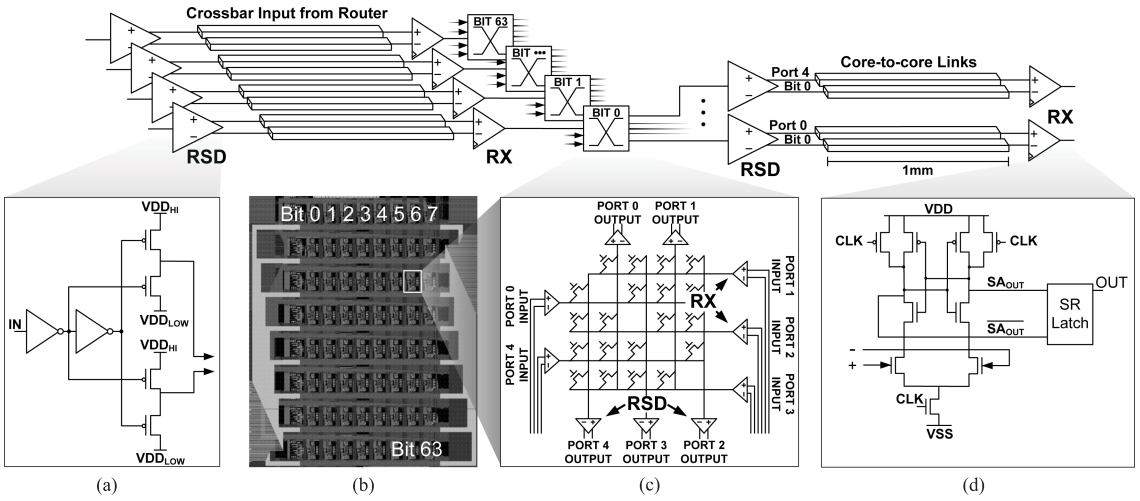


Fig. 7. Crossbar and link circuit implementation. (a) RSD. (b) Bit-slice array crossbar layout. (c) Crossbar bit-slice schematic with link drivers at slice outputs ports. (d) Clocked sense amplifier receiver (RX).

Differential low-voltage swing ($V_{SWING} = 0.45$ V) is used in [15], but several important differences with our proposed NoC are notable. First, V_{SWING} is 1.5× or more larger than this paper, such that receiver offset (due to process variation) and noise coupling from full-swing digital logic below are not problematic. Second, the wire-dominated crossbars in [15] do not utilize low-voltage swing techniques.

To reduce the energy required to drive large capacitive wire loads, reduced-voltage swing signaling was implemented using dual voltage supply, differential, reduced-swing drivers (RSD), Fig. 7(a), followed by a simple sense-amplifier receiver as shown in Fig. 7(d). The nominal chip supply voltage is used for the upper voltage supply while the lower supply voltage is generated off-chip. This allows for signal swings to be easily adjusted during testing as the voltage difference between the two supplies and sets the common mode voltage without requiring static power dissipating (except for leakage) in either the driver or receiver. In practice, a voltage supply 0.2 V–0.4 V below the core logic voltage is often already available on-chip for the SRAM caches [26] or other components that operate at a reduced supply voltage, and is therefore a small implementation overhead.

Using a second supply offers a number of advantages over single-supply, low swing signaling techniques. First, the energy required to drive the low-swing signal scales quadratically with voltage swing. Second, links are actively driven, making them less susceptible to crosstalk than capacitively-coupled wires, easing constraints on the link routing and any surrounding noisy environment. Finally, the dual supply drivers require no DC biasing of the common mode, compared with feed-forward capacitive coupling as in [25].

While differential signaling approximately doubles the wire capacitance of each bit by introducing a second wire, it removes the necessity of multiple inverter buffer stages for driving long wires and enables the use of reduced voltage swings, resulting in quadratic energy savings in the datapath. Thus, if the energy required to drive a single-ended full swing wire is given by (1), then the energy required to drive the differential wire pair at 200 mV is approximately given by (2).

$$E_{\text{swing}=1.2\text{ V}} = \frac{1}{2} C_{\text{wire}} \text{V}^2 \tag{1}$$

$$E_{\text{swing}=0.2\text{ V}} = \frac{1}{2}(2C_{\text{wire}})\frac{1}{36}\text{V}^2 = \frac{1}{18} E_{\text{swing}=1.2\text{ V}} \tag{2}$$

Hence, reducing the voltage swing from 1.2 V to 200 mV results in greater than 94% reduction in the energy required to drive the interconnect wire. The link pitch density is limited by the transmitter/receiver layout areas, such that bandwidth/mm is minimally changed for the differential wiring signaling.

The area-efficient sense amplifier receivers, shown in Fig. 7(d) are comprised of near-minimum sized transistors and exhibit approximately 100-mV simulated input offset across Monte Carlo simulations. Occupying 7.8 μm^2 and 15.2 μm^2, respectively, the same driver and receiver circuits are used in both the crossbar and link designs. Note for comparison, in the technology used, the smallest DFF standard cell available for a full swing implementation occupies 14.8 μm^2.

Further reductions in voltage swing requires either larger transistors or offset correction in the receiver in order to overcome the receiver input offset due to process variation. However, at 200-mV signal swing, the energy required to drive the wires accounts for only 2% of the energy consumed in the links and crossbar, with the rest being accounted for in the clock distribution, driver input capacitance, sense amplifier and in the case of the crossbar, the port selection signals. Therefore, the resulting increases in receiver area required to further reduce the voltage swing result in diminishing returns for improved energy efficiency.

Datapaths in the crossbar range in length from 150 μm to 450 μm. Differential wires are tightly packed and spaced apart by 0.28 μm, limited not by the minimal wire spacing but by the via size. Link lengths are asymmetric across different ports, with wire routes ranging from 65 μm to 1 mm in length. Each of the proposed NoCs routers contains 640 differential pairs and transceivers, making dense wiring, small transceiver sizes, and minimal energy/bit at network operating speed the primary design requirements.

B. Reduced-Swing Crossbar

The simplest and most obvious crossbar layout is to route a grid of vertical and horizontal wires with pass-gates or tri-state buffers at their intersection points, as shown in Fig. 7(c). While simple, this approach suffers from a number of major disadvantages, including poor transistor density, low bandwidth, and a bit-to-area relationship.

In practice, higher crossbar speeds and improved density can be achieved in a standard digital synthesis flow using mux-based switches that place buffered muxes throughout the area of the crossbar. For larger crossbars in particular, speed can be further improved by pipelining the crossbar traversal, allowing sections of the wire load to be driven in separate clock cycles. While simple to implement in digital design flows, both approaches introduce additional loading in the form of increased fan-out buffering and clock distribution that results in increased power consumption. Furthermore, network latency is also negatively impacted.

The crossbar implemented in our design improves energy efficiency by replacing crossbar wires with low-swing signaling circuits. This approach seeks to drive the large wire capacitances of the crossbar with a reduced voltage swing, without introducing additional buffers or clocked elements. Implemented as a bit-sliced crossbar, each of the 64-bits in each of the five input buses is connected to a one-bit wide, five-input to five-output crossbar. An 8×8 grid is then patterned out of 64 of these bit-cell crossbars in order to construct a 64-bit wide, 5×5 crossbar as shown in Fig. 7(b).

Each crossbar bit-slice consists of five Strongarm sense amplifiers receivers (RX), a 5×5 switching matrix (20 single-ended pass-gates), and five low-swing transmitters (RSD) at each bit-slice output, as shown in Fig. 7(c). Each of the five reduced-swing differential inputs is driven externally to the input of the crossbar by a RSD, which is connected to the output of the router logic. At the positive clock edge, each of the five low-swing differential inputs is first converted to full-swing logic by the sense amplifier, then driven through a short 6-μm wire via a single-ended pass-gate transistor controlled by the switch arbiter, and finally sent out of the selected output port via the interconnect RSD. In our routing scheme, U-turns are not allowed, so each of the five crossbar input ports can be assigned to one of four possible output ports.

The receiver acts as a sense-amplifier flip-flop with low-swing differential inputs, replacing the flip-flop that would otherwise reside at the output of the crossbar-traversal pipeline stage. Like mux-based crossbars, this crossbar topology results in irregular datapaths across the 64 b parallel interconnect, requiring that the maximum crossbar speed be bounded by the longest datapath delay through the crossbar.

Full-swing select signals are routed on M1 and M2, differential data signals are routed on minimum width wires on M3–M5, and a separate clock wire is routed on M7 for each port. The clock distribution path is routed to closely match the worst case RC delay of the datapath to match clock skews. The crossbar switch allocator also implements clock gating, activating only the crossbar receive port that is expected to be used for switch/LT.

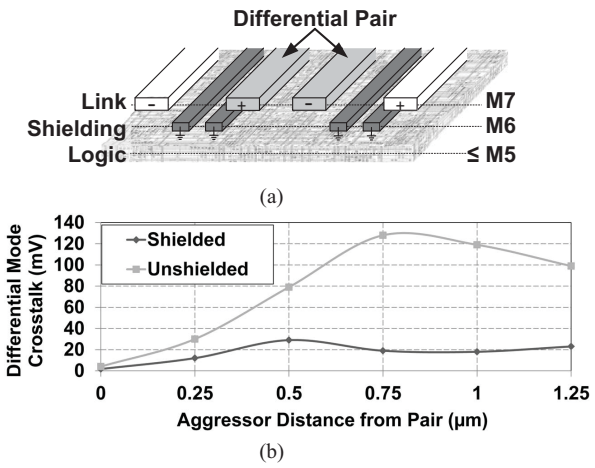


Fig. 8. (a) Layout of differential mode link shielding and (b) effectiveness of differential mode shielding at reducing crosstalk from full swing aggressor logic.

C. Differential Mode Shielding for Crosstalk Reduction

A major concern for reduced-swing signaling is the increased susceptibility of the low swing signals to crosstalk from a routers full-swing digital logic on lower metal layers. While differential signaling and adjacent wire twisting [27] are effective at rejecting common-mode crosstalk from nearby wires, care must be taken to minimize any asymmetric capacitive coupling from potential aggressor signals.

Complete ground plane shielding below the entire signal path establishes both a well-defined routing environment, and provides the best protection against full-swing digital coupling. Unfortunately, this conservative shielding environment also contributes additional capacitance to the already wire-dominated load.

An alternative approach used here is to route shielding on M6 between and in parallel to the differential pairs on M7 as shown in Fig. 8(a). In this manner, differential mode crosstalk which degrades both timing and signal margin by coupling asymmetrically onto one of the differential wires is significantly reduced. Common-mode crosstalk that couples equally onto both wires is intrinsically rejected by the differential input of the receiver. This approach can improve energy-efficiency by shielding only the differential mode crosstalk while reducing the per-unit length wire capacitance attributed to wire shielding. Thus, lower capacitance is achieved when compared with full plane ground shielding at the cost of greater common-mode coupling. Hence, the reduced-voltage swing can be decreased close to the minimum level obtained from complete ground shielding.

The cycle alloted to the crossbar and LT stages provides sufficient timing margin for the reduced-swing signals to settle even in the presence of crosstalk-induced delay. However, signal integrity of the reduced-swing wires after the value has settled, but immediately before it is sampled by the sense-amp quantizer is a greater concern. Extracted SPICE simulations were used to evaluate the worst-case crosstalk on a 1-mm differential link, comparing differential-mode shielding with no signal shielding. A 1 mm, full-swing aggressor signal was

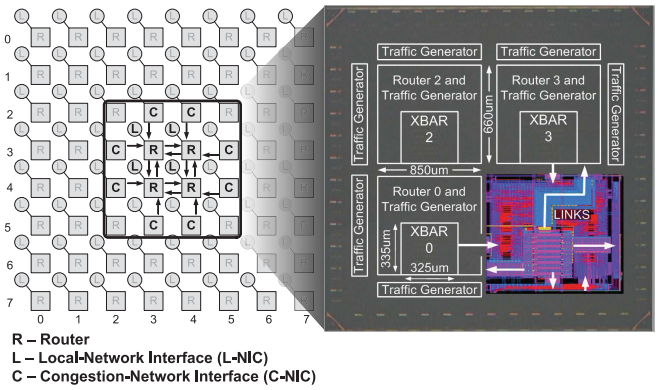


Fig. 9. 2×2 SWIFT network prototype overview and die photo overlaid with node 1 layout.

routed on M5 in parallel to the differential wires of the link on M7, with shielding inserted on M6. The aggressor crosstalk was measured by sweeping the aggressor laterally from the center of the differential pair (where all the crosstalk appears as common-mode), up to a distance of 1.25 μm from the center of the pair. Fig. 8(b) shows a 4.4$\times$ reduction in the worst-case aggressor noise coupling using the proposed differential-mode shielding.

Maxwell 2-D field-solver was used to more accurately predict the effectiveness of differential-mode shielding at reducing capacitance on the signal line. When modeled with the 90-nm process specification, this approach shows a 19% reduction in capacitance on the signal wires when compared with a complete grounded plane design. If the signal swing is increased to compensate for the conservative, worst-case differential mode crosstalk estimation of 29 mV from the strongly-driven, 1-mm parallel aggressor modeled in Fig. 8(b), the differential mode shielding yields a net energy savings when the signaling voltage swing is below 250 mV.

V. Results

In this section, we report both the simulated and measured results of the SWIFT NoC prototype, and compare it to a baseline NoC.

A. SWIFT NoC

The SWIFT NoC parameters are shown in Table I. We chose eight buffers per port, shared by 2 VCs. This is the minimum number of buffers required per port, with one buffer reserved per VC for deadlock avoidance and six being the buffer turnaround time with on–off signaling between neighboring routers. We used standard-cell libraries provided by ARM Corporation for synthesis. The place and route of the router RTL met timing closure at 600 MHz. The process technology used and use of standard cells instead of custom layouts, limits our router design from running at GHz speeds, such as in [3]. Note that based on extracted layout simulations, the custom reduced-swing transceivers are designed to operate at 2 GHz across 1-mm distances with 250-mV voltage swing.

We fabricated a 2×2 slice of our 8×8 mesh, as shown in Fig. 9. We added on-chip pseudo-random traffic generators at all local four network interfaces and at the eight unconnected ports at the corners of the mesh, resulting in a total of 12 traffic generating NICs.

In an actual chip-multi processor, a tile consists of both the processing core and a router, with the router accounting for approximately a quarter of the tile area [3]. Since we did not integrate processing cores in this design, we handle to place the routers in order to conserve area. This results in asymmetric link lengths in our chip, with drivers sized for the worst-case of 1-mm links. A photo of our prototype test-chip overlaid with the layout of node 1 is shown in Fig. 9.

Due to the 4 mm^2 network size, we used a synchronous clock rather than a globally-asynchronous, locally-synchronous approach as in [3], which was outside the scope of this paper. The test chip operates at 400 MHz at low load, and 225 MHz at high injection rates with a supply of 1.2 V. We found that the performance of the test chip was limited from achieving higher clock speeds due to resistive drops in the power routing grid that were not caught prior to fabrication.

B. Baseline NoC

To characterize the impact of the various features of the SWIFT NoC, we implemented a baseline VC router in the same 90-nm technology.

A VC router needs to performs the following actions: buffer the incoming flit (buffer write or BW), choose the output port (route compute or RC), arbitrate and choose an input VC winner for each input port of the switch (switch allocation-inport or SA-I), arbitrate and choose an input port winner for each output port of the switch (switch allocation-outport or SA-O), arbitrate and choose a VC for the next router (VC allocation or VA), read winning flits out of the buffer (buffer read or BR) and finally send the winning flits through the crossbar switch (ST) to the link connecting to the next router/NIC.

We design our baseline router similar to the UT TRIPS [2] and Intel TeraFLOPS [3] routers which use VCs. We design a three-stage router pipeline, the details of which are given in the Appendix. We leverage recent research in shared input buffers [5], lookahead routing [22], separable allocation [28], and VC selection instead of full VC allocation [11] allowing us to optimize the design heavily and perform many of the router actions, which were discussed earlier, in parallel. A target frequency specification of 600 MHz or more restricted by 90-nm standard cells led us to the three-stage baseline design. UT TRIPS [2] also uses a three-stage router pipeline operating at 366 MHz. Intel TeraFLOPS [3] uses a five-stage router pipeline (and an additional stage in the link), but is able to operate at 5 GHz[3] due to custom blocks instead of standard-cells. The SWIFT NoC is the first NoC prototype demonstrating a one-cycle router pipeline in silicon that bypasses buffering completely to save power. The Tilera TILEPro64 [4] uses five separate networks, instead of using VCs. As a result, it does not need to perform SA-I or VA.

[3]Theoretically, Intel's router could perform all operations within one-cycle for operating frequencies less than 1 GHz, but flits would still have to get buffered (and read out the same cycle upon successful arbitration).

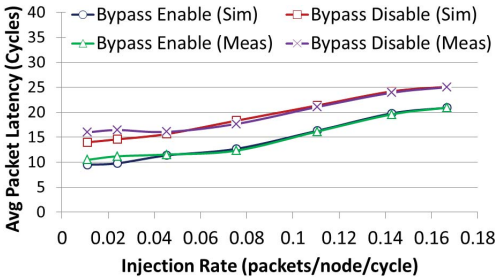


Fig. 10. Network performance results for fabricated 2 × 2 chip.

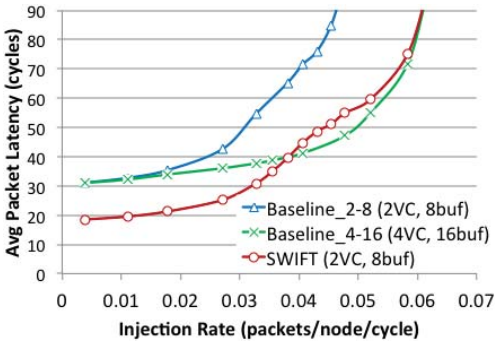


Fig. 11. Network performance results in cycles for 8 × 8 networks.

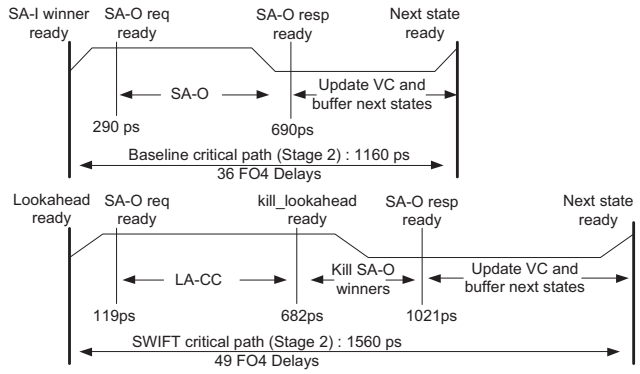


Fig. 12. Critical paths of the SWIFT router and baseline router.

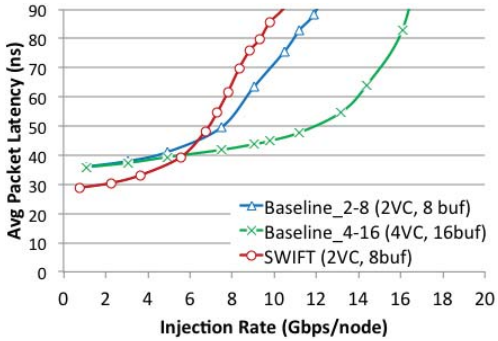


Fig. 13. Network performance results in ns for 8 × 8 networks.

In addition, an XY-routing scheme allows the router to present a one-cycle delay to flits going straight, and two-cycles to flits that are turning. TILEPro64 s design philosophy of using physical instead of VCs is a research topic in itself, and thus comparing SWIFT to it quantitatively is beyond the scope of this paper.

The nonbypass pipeline in SWIFT is the same as the baseline pipeline, thus allowing us to compare the performance, power, and area of the SWIFT and the baseline designs and the impact of our additions (bypass-logic and the reduced-swing crossbar).

Once we finalized the baseline router pipeline, we swept the number of VCs and buffers in the baseline such that the peak operating throughput of both the baseline and the SWIFT NoC was the same. This is described in Section V-C. We used two of the configurations for power comparisons, which are described in Section V-D.

C. Network Performance

Fig. 10 demonstrates that the measured and simulated latency curves match, confirming the accuracy and functionality of the chip. The 2 × 2 network delivers a peak throughput of 113 bits/cycle.

The primary focus of this paper was to implement the TFC router in silicon, and integrate reduced-swing circuits in the datapath. Thus, we do not perform a full performance analysis of the design across different traffic patterns. The original TFC paper [8] evaluates TFC for synthetic uniform-random, tornado, bit-complement, and transpose traffic. It also evaluates the impact of TFC with real-world application traces from the SPLASH benchmark suite. In this paper, we evaluate the latency-throughput characteristics of the SWIFT NoC with uniform random traffic via on-chip traffic generators to validate the chip functionality. We also use this analysis to set the parameters of an equivalent baseline router (same operating throughput) for a fair comparison of power.

1) Average Packet Latency (Cycles): We first compare the average packet latencies of the 8 × 8 SWIFT NoC and the baseline NoC in cycles via RTL simulations. Fig. 11 plots the average packet latency as a function of injection rate for SWIFT, and two interesting design points of the baseline: Baseline_2–8 (2VC, 8 buffers) and Baseline_4–16 (4VC, 16 buffers). At low loads, SWIFT provides a 39% latency reduction as compared to the baseline networks. This is due to the almost 100% successful bypasses at low traffic. At higher injection rates, Baseline_2–8 saturates at 21% lower throughput. SWIFT leverages adaptive routing via tokens, and faster VC and buffer turnarounds due to bypassing, in order to improve link utilization which translates to higher throughput. Baseline_4–16 matches SWIFT in peak saturation throughput (the point at which the average network latency is three times the no-load latency) in bits/cycle.

2) Average Packet Latency (ns): Fig. 12 shows that the critical paths of the SWIFT and the baseline routers, which occur during the SA-O stage in both designs, amount to 49 and 36 FO4 delays, respectively. We observe that the baseline is 400 ps faster, and therefore a dissection of the various components of the critical path provides interesting insights. The primary bottleneck in the SWIFT microarchitecture occurs when SA-O winners need to be quenched, exhibiting an additional 339 ps of extra delay. Note that the SWIFT router was designed to perform the SA-O and LA-CC stages in

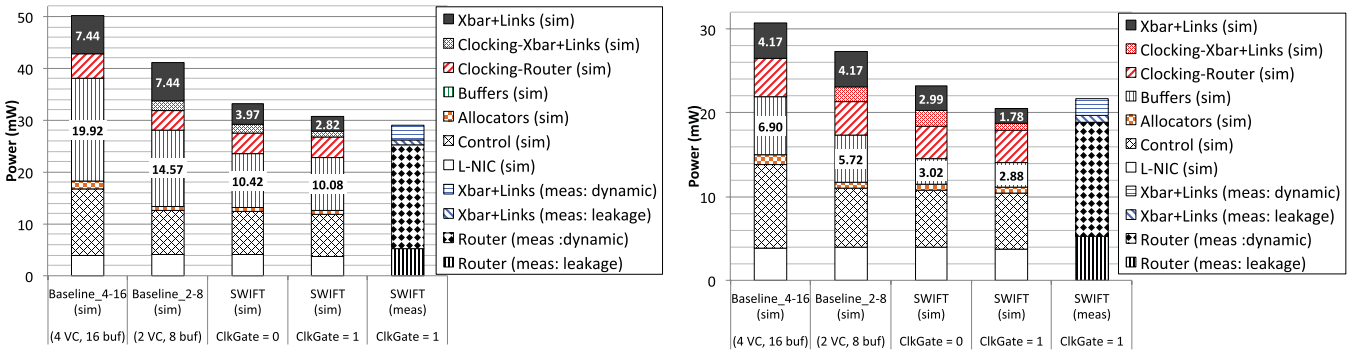


Fig. 14. Tile power at (a) high traffic injection (1 packet/NIC/cycle) and (b) low traffic injection (0.03 packets/NIC/cycle) rates.

parallel, followed by the removal of SA-O assignments in case they conflicted with the lookahead assignments for the crossbar, in order to maintain higher priority for the lookaheads. In hindsight, if we had allowed both the lookahead and local VC requests to move to the same switch arbiters, relaxing lookahead priority, the critical path would have been significantly reduced.

If we take these critical paths in account, the baseline network can run at a frequency 1.34 times faster than SWIFT. Under this operating condition, Fig. 13 shows the performance results of the 8×8 NoCs in nanoseconds, instead of cycles. The SWIFT NoC shows a 20% latency reduction at low-load as compared to the baselines, and similar saturation throughput as Baseline_2–8.

D. Power

We compare the SWIFT and baseline routers at the same performance (throughput) points for fairness. In Section V-C, we observed that Baseline_4–16 matches SWIFT in saturation throughput if both networks operate at the same frequency. Baseline_2–8 matches SWIFT in saturation throughput if it operates at a higher frequency, or if the networks are operating at low loads. We report power numbers for both Baseline_2–8 and Baseline_4–16 for completeness.

We perform power simulations and measurements at a frequency of 225 MHz and VDD of 1.2 V, and the results are shown in Fig. 14(a) and (b) at high and low loads, respectively. In both graphs, all 12 traffic generator NICs are injecting traffic. The low-swing drivers were set to 300-mV signal swing. Because the L-NIC shares a supply with the router while the crossbar shares a supply with the reduced-swing links, it was not possible to measure each of the blocks separately. Instead, post-layout extracted simulations were performed to obtain an accurate relative breakdown of the power consumption of the different components, which were then compared and validated with chip measurements of the combined blocks.

At high loads, operating at the same frequency, Baseline_4–16 matches SWIFT in performance, but has 49.4% higher buffer (control path) and 62.1% higher crossbar and link (datapath) power. SWIFT [last two bars in Fig. 14(a)] achieves a total power reduction of 38.7% at high injection, with the chip consuming a peak power of 116.5 mW.

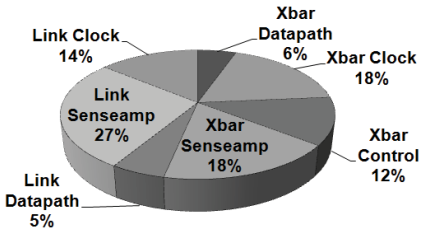


Fig. 15. Contributions to datapath energy at network saturation.

At low loads, operating at the same frequency, Baseline_2–8 can match SWIFT in performance, but consumes 24.6% higher power than SWIFT [last two bars in Fig. 14(b)].

Baseline_2–8 and SWIFT have the same VC and buffer resources. SWIFT adds buffer bypassing logic (using tokens and lookaheads), and the low-swing crossbar. Thus comparing Baseline_2–8 and the first bar of SWIFT shows us that buffer bypassing reduces power by 28.5% at high loads, and 47.2% at low loads, while the low-swing datapath reduces power by 46.6% at high loads and 28.3% at low loads. These results are intuitive, as buffer write/read bypasses have a much higher impact at lower loads when their success rate is higher, while datapath traversals are higher when there is more traffic.

Lookahead signals allow the crossbar allocation to be determined a cycle prior to traversal, making per-port, cycle-to-cycle clock gating possible. Therefore, clock gating was implemented at each crossbars input port, using the crossbars forwarded clock, reducing the crossbar clock distribution power by 77% and 47%, and sense amplifier power by 73% and 43% at low and high injection, respectively.

The combined average energy-efficiency of the crossbar and link at the network saturation point is measured to be 128 fJ/bit, based on chip measurements of the crossbar and link currents, and the count of the received packets. This value is further broken down into component contributions in Fig. 15.

1) Link and Crossbar Circuits: Fig. 16 shows the distribution of energy consumption in the driver, wire, and receiver of a 1-mm link as a function of wire signaling voltage swing. Energy/bit/mm values are averaged across 10 000 cycles of random data at 2 GHz.

When not limited by the routers critical path, the reduced-swing transceivers are operational across 1-mm wires at

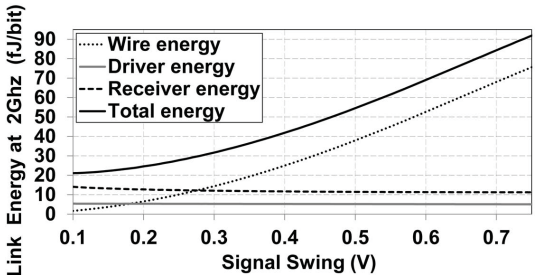


Fig. 16. Circuit contributions to link energy.

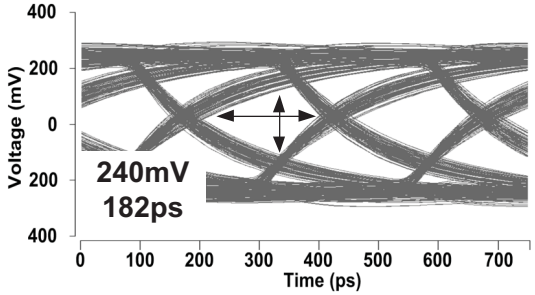


Fig. 17. Post-layout simulated eye at output of 1-mm link showing 52% eye opening at 2 GHz.

2 GHz with a 250-mV signal swing (post-layout simulations), achieving a theoretical peak throughput of 640 Gb/s per crossbar. From post-layout simulations, 28 fJ/bit is observed for transmission across 1-mm links, including RSD input capacitance, wire energy, and sense amplifier. Fig. 17 shows 52% eye closure (240 mV) at the sense amplifier input of the 1-mm link, representing approximately the worst RC delay-path observed in the fabricated chip. A comparison with previous interconnect works is summarized in Table II.

2) Overheads: The west-first adaptive routing logic used for tokens, the lookahead arbitration logic, and the bypass muxes account for less than 1% of the total power consumed in the router, and are therefore not a serious overhead. This is expected, as the allocators account for only 3% of the total power, consistent with previous NoC prototypes. The control power of the SWIFT NoC is observed to be 37.4% lower than Baseline_4–16, due to fewer buffers and VCs (hence smaller decoders and muxes) required to maintain the same throughput.

The overall control power of SWIFT is approximately 26% of the entire router power, as seen in Fig. 14. This high proportion is primarily due to a large number of flip-flops in the router, many of which were added conservatively to enable the design to meet timing constraints, and could have been avoided by using latches. In addition, the shared buffers require significant state information in order to track the free buffer slots and addresses needed for each flit, adding more flip-flops to the design.

E. Area

The baseline and SWIFT routers primarily differ in the following hardware components: tokens, lookahead signals with corresponding bypassing logic, buffers, VCs, and crossbar implementation. We estimate the standard-cell area contributions for each of these components and compare them

TABLE II
SUMMARY OF INTERCONNECT TRANSCEIVERS

	[13]	[25]	Inverter and Flip-flop	Simulated link	Measured crossbar and Link
Process	90 nm	90 nm	90 nm	90 nm	90 nm
Data rate	4 Gb/s	9 Gb/s	1 Gb/s	2 Gb/s	225 MHz
Link length	10 mm	2 mm	1 mm	1 mm	~2 mm
TX Area	1120 μm^2	C = 7.29 μm^2 W_P = 1.6 μm $W_N = 4\mu$	6.35 μm^2	7.8 μm^2 W_P = 2.4 μm	2 × 7.8 μm^2
RX area	1760 μm^2	Not available	14.8 μm^2	15.2 μm^2	2 × 15.2 μm^2
Signal swing	~200 mV	120 mV	1.2 V	250 mV	250 mV
Energy/bit	356 fJ	105 fJ	305 fJ	28 fJ	128 fJ

TABLE III
AREA COMPARISON (ABSOLUTE AND PERCENTAGE)

Component	SWIFT Area, % of total	Baseline Area, % of total
Tokens	1235 μm^2, 0.82%	0
Bypass	10 682 μm^2, 7.10%	0
Buffers	72 118 μm^2, 47.94%	81 231 μm^2, 40.08%
Crossbar	15 800 μm^2, 10.50%	21 584 μm^2, 10.64%
Control	50 596 μm^2, 33.63%	99 856 μm^2, 49.27%
Total	150 431 μm^2, 100%	202 671 μm^2, 100%

in Table III. For the custom crossbar, we use the combined area of the RSD, transistor switching grid, and sense amplifier circuits as the metric to compare against the matrix crossbars cell area. The total area of the SWIFT router is 25.7% smaller than the Baseline_4–16. This is an important benefit of the SWIFT design: the 8% extra circuitry, required for implementing tokens and bypassing, results in a 11.2% reduction in buffer area and 49.3% reduction in control area (due to fewer VCs and corresponding smaller decoders and allocators) required to maintain the same peak bandwidth, thereby reducing both area and power.

Note that the SWIFT NoC exhibits some wiring overheads. The 23-bit token signals from the three-hop neighborhood at each router add 7% extra wires per port compared to the 64-bit datapath. The 14 lookahead bits at each port carry information that is normally included in data flits, and are therefore not strictly an overhead. To lessen this overhead, the flit width could have either been shrunk or packets could have been sent using fewer flits, which would further enhance SWIFTs area and performance benefits over the baseline. Finally, while Table III highlights that the active device area of the reduced-swing custom crossbar is less than that of a synthesized design, differential signaling requires routing twice as many wires as well as potentially requiring an additional metal layer beneath, if shielding is required to mitigate digital crosstalk from below.

VI. INSIGHTS

In this section, we discuss some of the insights gained from this paper.

A. Tradeoffs

The immediate tradeoff of TFC versus the baseline that is visible from the results of this paper is the critical path. However, we believe that the TFC critical path can be shrunk further by a more optimized LA-CC. A relaxed priority for lookaheads over local requests, or a more optimized priority arbiter would have helped us to reduce the 339 ps of clear overhead.

The proportion of control power in our router is another aspect that requires additional work. Fig. 14 highlighted that about 26% of the router power is in the control, which primarily consists of state within the VCs and the input port. The shared buffer is a major contributor: 1) every VC needs to track where its 5 flits are buffered and 2) a linked list of free buffers has to be maintained. While we clock-gated the crossbar, all flip flops at the routers were still active irrespective of traffic. Adding clock-gating/power-gating within the router will also help to reduce the control power.

In our implementation of TFC [8], we implemented normal tokens. Normal tokens only perform speculative bypass and require VC reservation at every hop to account for the lookahead getting killed and forcing buffering. Guaranteed tokens from the TFC design allow flits to perform a guaranteed bypass and enhance throughput further by enabling flits to bypass intermediate routers that do not have any free buffers. We plan to incorporate this aspect in future work.

B. Technology Projections

We used a 90-nm process in this paper, which is about two generations away from the current state-of-the-art technology. Moving to a smaller technology node impacts both the microarchitectural aspects of this design as well as the circuits.

In terms of microarchitecture, a newer technology will help to shrink the critical path. A synthesis of the same design in IBM 45 nm resulted in a critical path of 1ns as opposed to 1.56 ns in 90 nm. A faster technology node thus enables an intricate pipeline like TFC to be realizable.

Moving to smaller process nodes introduces additional considerations for the interconnect and low-swing circuits. Interconnect energy continues to scale more slowly than logic energy as core-to-core wire lengths remain relatively static and on-chip communication requirements increase, resulting in an increasing benefit for implementing area efficient low-swing interconnect driver and receiver circuits. At the same time, the number of low-swing interconnects may increase dramatically, resulting in the manual implementation of these circuits to require an unreasonable amount of design effort. Automated implementation and validation of these low-swing interconnects will become critical in order to keep up with the pace of digital design flows, and is an ongoing area of research.

Finally, device variation is projected to worsen, requiring that sense amplifier circuits either be increased in size relative to logic or implement a compact input offset calibration, [29], in order to provide sufficient sensitivity to low-voltage input signals without bloating area.

VII. Conclusion

In this paper, we presented a NoC that utilizes low-power architecture and circuit co-design to improve the power, latency, and throughput of a NoC. In particular, a token-based smart pipeline bypassing scheme, and a reduced-swing crossbar and interconnect together contribute to latency and power improvements in an 8×8 network running uniform random traffic, while requiring half as many buffers as extracted simulations of a baseline NoC using virtual-channel routers. Under uniform random traffic, a reduction of 38% in peak network power was reported when networks were operated at identical frequency conditions, while a 20% reduction in low-load latency was reported when both networks are run at their maximum operating frequencies. Reduced swing circuits achieve 62% power savings in the datapath versus a full-swing, synthesized implementation. Differential mode shielding was also presented as a means to enable protected, reduced-swing signaling over digital logic with less capacitive loading than full ground plane shielding. Many of the architectural and circuit novelties in SWIFT would enhance any NoC router/link design, as SWIFT performs more efficient allocation of network links and buffers, enabling low-power traversal. We hope this paper paves the way for more such prototype designs. Demonstrating a SWIFT-like NoC design on a multicore chip with real application traffic is part of our future work.

Appendix

Baseline Nonbypass Pipeline Microachitecture

The actions performed by NoC routers are buffer write/read, routing, switch allocation, VC allocation, and switch TRAVERSAL. These actions can be pipelined into multiple stages based on implementation decisions, and target frequencies. We implement a highly optimized three-stage pipeline that builds on the simpler routing architectures in UT TRIPS [2] and Intel TeraFLOPS [3] by leveraging research in shared input buffers [5], lookahead routing [22], separable allocation [28], and parallel SA/VA [11], [28]. We use this both for our baseline router, and as the nonbypass pipeline in the SWIFT router. These stages are shown in Fig. 4(a). Flits in the SWIFT NoC traverse this pipeline if bypassing fails due to one of the paths shown in the flow chart of Fig. 6.

A. Stage 1-BW

Incoming flits are written into buffers at each input port, which were implemented with register files generated from the foundry memory generators. These input buffers are organized as a shared pool among multiple VCs [5]. The addresses of the buffers are connected as a linked list. An incoming flit that requires a free buffer obtains the address from the head of the linked list, and every buffer that is freed up appends its address to the tail of the linked list. One buffer is reserved per VC in order to avoid deadlock. Compared to private buffers per VC, which can be implemented as a FIFO, our shared buffer design incurs an overhead of storing the read addresses of all flits in the VC state table, but has the advantage of reducing the numbers of buffers required at each port to satisfy buffer

turnaround time (minimum number of cycles before which the same buffer can be reused).

B. Stage 1-SA-I

An input VC is selected from each input port to place a request for the switch. This is implemented using V:1 round robin arbiters at each input port, where V is the number of VCs per port. Round robin arbiters are simple to implement [5] and ensure that every VC gets a chance to send a flit.

C. Stage 2-SA-O

The winners of SA-I at each input port place requests for their corresponding output ports. As no u-turns are allowed, there can be a maximum of four input ports requesting the same output port. These conflicts are resolved using 4:1 matrix arbiters, one for each output port. Matrix arbiters are used for fair allocation of the crossbar output port to all input ports [5]. Separating switch allocation into two phases of simpler arbitration, SA-I and SA-O, is a common approach to satisfy minimum cycle time constraints [28]. Note that a flit may spend multiple cycles in switch allocation due to contention.

D. Stage 2-VC Allocation (VA)

At the end of SA-O, winning head flits are assigned an input VC for their next hop. (Body and Tail flits follow on the same VC). VC allocation in our design is a simple VC selection scheme, based on [11]. Each output port maintains a queue of free VCs at the input port of the next router. A switch request is allowed to be placed for an output port only if the router connected to that output port has at least one-free input VC. The winning head flit of a switch output port, at the end of SA-O, picks up the free VC at the head of the queue and leaves. Thus, there is no full-fledged arbitration required, simplifying the VC allocation process. If the router receives a signal indicating a free VC from the next router, the corresponding VC id is enqueued at the tail of the queue. VA does not add any extra delay to the critical path since the updating of the queue and the computation of the next free VC id take place in parallel to SA-O.

E. Stage 2-BR

Flits that won SA-I start a pre-emptive read of the buffers, in parallel to SA-O. This is because the register files require all input signals to be ready before the clock edge. If we wait until SA-O declares the winner of the switch output port, BR would have to be pushed to the next cycle, adding latency. The drawback of this is that there are wasted reads from the buffer which would consume power. We address this by biasing SA-I to declare the same input VC as the winner until it succeeds to use the crossbar. This ensures that the same address is read out of BR to avoid additional switching power.

F. Stage 3-ST

The flits that won the switch ports traverse the crossbar switch.

G. Stage 4-LT

The flits coming out of the crossbar traverse the links to the next routers.

ACKNOWLEDGMENT

The authors would like to thank the Trusted Access Program Office, Simi Valley, CA, for fabrication of our test chip, and ARM for providing standard cell libraries. They would also like to thank A. Kumar from Intel Corporation, Santa Clara, CA, for help in designing the SWIFT pipeline, K. Hu from Broadcom Corporation, Irvine, CA, for assisting in post-layout verification, and the reviewers for their patience and detailed feedback.

REFERENCES

[1] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, J.-W. Lee, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The raw microprocessor: A computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar.–Apr. 2002.

[2] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S. W. Keckler, and D. Burger, "On-chip interconnection networks of the TRIPS chip," *IEEE Micro*, vol. 27, no. 5, pp. 41–50, Sep.–Oct. 2007.

[3] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep.–Oct. 2007.

[4] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown, III, and A. Agarwal, "On-chip interconnection architecture of the Tile processor," *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep.–Oct. 2007.

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

[6] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. 31st Annu. Int. Symp. Comput. Arch.*, Jun. 2004, pp. 1–188.

[7] H. Matsutani, M. Koibuchi, H. Amano, and T. Yoshinaga, "Prediction router: Yet another low latency on-chip router architecture," in *Proc. High-Perform. Comput. Arch.*, Feb. 2009, pp. 367–378.

[8] A. Kumar, L.-S. Peh, and N. K. Jha, "Token flow control," in *Proc. 41st Int. Symp. Microarch.*, Nov. 2008, pp. 342–353.

[9] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Toward the ideal interconnection fabric," in *Proc. 34th Annu. Int. Symp. Comput. Arch.*, Jun. 2007, pp. 150–161.

[10] L. Xin and C. S. Choy, "A low-latency NoC router with lookahead bypass," in *Proc. Int. Symp. Circuits Syst.*, Jun. 2010, pp. 3981–3984.

[11] A. Kumar, P. Kunduz, A. P. Singhx, L.-S. Pehy, and N. K. Jhay, "A 4.6 Tbits/s 3.6 GHz single-cycle NoC router with a novel switch allocator in 65 nm CMOS," in *Proc. 25th Int. Conf. Comput. Design*, Oct. 2007, pp. 63–70.

[12] J. Bae, J.-Y. Kim, and H.-J. Yoo, "A 0.6 pJ/b 3 Gb/s/ch transceiver in 0.18 μm CMOS for 10 mm on-chip interconnects," in *Proc. IEEE Int. Symp. Circuit Syst.*, May 2008, pp. 2861–2864.

[13] B. Kim and V. Stojanovic, "A 4Gb/s/ch 356 fJ/b 10 mm equalized on-chip interconnect with nonlinear charge-injecting transmit filter and transimpedance receiver in 90 nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 2009, pp. 66–68.

[14] D. Schinkel, E. Mensink, E. A. M. Klumperink, E. van Tuijl, and B. Nauta, "A 3-Gb/s/ch transceiver for 10-mm uninterrupted RC-limited global on-chip interconnects," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 297–306, Jan. 2006.

[15] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-performance SoC design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 2, pp. 148–160, Feb. 2006.

[16] H.-S. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proc. 36th Annu. IEEE/ACM Int. Symp.*, Dec. 2003, pp. 105–116.

[17] L. Shang, L.-S. Peh, and N. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc. High-Perform. Comput. Arch.*, Feb. 2003, pp. 91–102.

[18] C.-H. Chen, S. Park, T. Krishna, and L.-S. Peh, "A low-swing crossbar and link generator for low-power networks-on-chip," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2011, pp. 779–786.

[19] A. Kodi, A. Louri, and J. Wang, "Design of energy-efficient channel buffers with router bypassing for network-on-chips (NoCs)," in *Proc. Int. Symp. Qual. Electron. Design*, Mar. 2009, pp. 826–832.

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

[21] G. Michelogiannakis, J. Balfour, and W. Dally, "Elastic-buffer flow control for on-chip networks," in *Proc. High-Perform. Comput. Arch.*, Feb. 2009, pp. 151–162.

[22] M. Galles, "Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER chip," in *Proc. High-Perform. Interconn.*, Aug. 1996, pp. 1–6.

[23] J. Oh, M. Prvulovic, and A. Zajic, "TLSync: Support for multiple fast barriers using on-chip transmission lines," in *Proc. 38th Annu. Int. Symp. Comput. Arch.*, 2011, pp. 105–116.

[24] T. Krishna A. Kumar, P. Chiang, M. Erez, and L.-S. Peh, "NoC with near-ideal express virtual channels using global-line communication," in *Proc. High-Perform. Interconn.*, 2008, pp. 11–20.

[25] D. Schinkel, E. Mensink, E. Klumperink, E. van Tuijl, and B. Nauta, "Low-power, high-speed transceivers for network-on-chip communication," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 1, pp. 12–21, Jan. 2009.

[26] S. Rusu, S. Tam, H. Muljono, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Vora, "A 45 nm 8-core enterprise Xeon® processor," *IEEE J. Solid-State Circuits*, vol. 45, no. 1, pp. 7–14, Jan. 2010.

[27] E. Mensink, D. Schinkel, E. Klumperink, E. van Tuijl, and B. Nauta, "Optimally-placed twists in global on-chip differential interconnects," in *Proc. Eur. Solid-State Circuits Conf.*, Sep. 2005, pp. 475–478.

[28] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. High-Perform. Comput. Arch.*, Jan. 2001, pp. 255–266.

[29] J. Postman and P. Chiang, "Energy-efficient transceiver circuits for short-range on-chip interconnects," in *Proc. Custom Integr. Circuits Conf.*, Sep. 2011, pp. 1–4.

Jacob Postman (S'10) received the B.S. degree in electrical and electronics engineering and computer engineering from Oregon State University, Corvallis, in 2008, where he is currently pursuing the Ph.D. degree in computer engineering.

His current research interests include energy-efficient on-chip interconnect circuits and their encompassing systems.

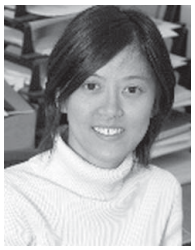
Mr. Postman was a recipient of the NSF Graduate Research Fellowship.

Tushar Krishna (S'08) received the B.Tech. degree from Indian Institute of Technology Delhi, New Delhi, India, and the M.S.E. degree from Princeton University, Princeton, NJ, in 2007 and 2009, respectively, both in electrical engineering. He is currently pursuing the Ph.D. degree with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge.

His current research interests include computer architectures and on-chip networks.

Christopher Edmonds received the B.S. degree in electrical and electronics engineering and computer science from Oregon State University, Corvallis, in 2009. He is currently pursuing the M.S. degree in computer science with Stanford University, Stanford, CA.

He is currently engaged in research with Microsoft.



Li-Shiuan Peh (S'99–M'01) received the B.S. degree in computer science from the National University of Singapore, Singapore, in 1995, and the Ph.D. degree in computer science from Stanford University, Stanford, CA, in 2001.

She has been an Associate Professor of electrical engineering and computer science with the Massachusetts Institute of Technology, Cambridge, since 2009. From 2002 to 2009, she was with Princeton University, Princeton, NJ. Her current research interests include low-power on-chip networks, parallel computer architectures, and mobile computing.

Dr. Peh was a recipient of the ACM Distinguished Scientist Award in 2011, the CRA Anita Borg Early Career Award in 2007, the Sloan Research Fellowship in 2006, and the NSF CAREER Award in 2003. She is a member of ACM.

Patrick Chiang (S'99–M'04) received the B.S. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 1998, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 2001 and 2007, respectively.

He is currently an Associate Professor of electrical engineering and computer science with Oregon State University, Corvallis. He is a Visiting Professor of developing wireless biomedical sensors with Fudan University, Shanghai, China. His current research interests include energy-efficient circuits and systems, including near-threshold computing, low-power serial link interfaces, and energy-constrained medical sensors.

Dr. Chiang was a recipient of the Department of Energy Early CAREER Award in 2010 and a NSF-CAREER Award in 2012. He is an Associate Editor of the IEEE TRANSACTIONS ON BIOMEDICAL CIRCUITS AND SYSTEMS, and on the Technical Program Committee of the IEEE Custom Integrated Circuits Conference.