Express Virtual Channels with Capacitively Driven Global Links

NETWORKS ON CHIP MUST DELIVER HIGH BANDWIDTH AT LOW LATENCIES WHILE KEEPING WITHIN A TIGHT POWER ENVELOPE. USING EXPRESS VIRTUAL CHANNELS FOR FLOW CONTROL IMPROVES ENERGY-DELAY THROUGHPUT BY LETTING PACKETS BYPASS INTERMEDIATE ROUTERS, BUT EVCS HAVE KEY LIMITATIONS. NOCHI (NOC WITH HYBRID INTERCONNECT) OVERCOMES THESE LIMITATIONS BY TRANSPORTING DATA PAYLOADS AND CONTROL INFORMATION ON SEPARATE PLANES, OPTIMIZED FOR BANDWIDTH AND LATENCY RESPECTIVELY.

Tushar Krishna
Amit Kumar
Li-Shiuan Peh
Princeton University

Jacob Postman
Patrick Chiang
Oregon State University

Mattan Erez
University of Texas at Austin

To achieve the higher performance demanded by next-generation applications, and to overcome the diminishing returns of complex uniprocessor designs, future microprocessors will need an increasing number of processor cores. As the number of cores increases, the performance of the network on chip (NoC) connecting them becomes critical. Traditional shared-bus architectures typically don't scale effectively to these large core counts. For example, multicore processors such as IBM's Cell Broadband Engine and Intel's Larrabee use a multihop NoC with a ring topology, whereas Intel's Teraflops processor, an 80-core research prototype, uses a mesh network. Incorporating a packet-switched fabric within a chip to interconnect the processor cores places strict power and area constraints on the NoC routers while requiring sustained bandwidth and short packet delivery latencies.

Our NoC with hybrid interconnect (Nochi) design uses multiple interconnect circuit types to improve latency while simultaneously reducing power. Nochi consists of a *data plane* for carrying high-bandwidth data payloads, and a *control plane* for providing timely control information to improve network efficiency. The data plane uses state-of-the-art on-chip routers with dense, high-bandwidth, full-swing links to provide the required network bandwidth. The control plane consists of ultralow-latency, multidrop on-chip global interconnect lines (G-lines), which provide instantaneous global information to the routers and enable a flow-control technique that significantly reduces router power while improving delivery latency. We optimized the control plane's design for latency and exchange of control information via broadcast. Hence, it only supports limited bandwidth and communication patterns and is unsuitable for carrying data directly.

Typical NoC routers are designed with four- to five-stage pipelines that add delay to network packets at every hop. The express virtual channel (EVC)[1] network control optimization technique sets up virtual express paths in the network that let packets bypass buffering and arbitration within a single dimension of the on-chip routers, thus approaching the latency and power characteristics of

Published by the IEEE Computer Society

Express virtual channels

In current state-of-the-art packet-switched on-chip networks, packets must compete for resources while going through a typically four- to five-stage router pipeline at each hop.[1] These routers dominate packet energy and delay. In the EVC[2] approach, flits can bypass some of the pipeline stages. The virtual channels at each port are partitioned into 1-hop, 2-hop, . . . k-hop EVCs. Flits that acquire a k-hop EVC can bypass $(k - 1)$ routers before being buffered again. Flits can acquire these EVCs during the pipeline's virtual channel allocation stage. Once a flit gets an EVC, it sends a lookahead one cycle in advance so the next router can set up the switch for this express flit. EVC flits are sent straight through the switch as soon as they reach an intermediate router, without any buffering or arbitration. These express flits are given higher priority over any local flits that might wish to use the particular switch ports during that cycle. Thus in the EVC design, flits reach their destinations faster and consume less power due to the bypassing.

Because on-chip networks must be lossless, an upstream node can send flits to a downstream node only if the flit is guaranteed a free buffer slot. Conventional routers exchange this information between adjacent routers using techniques such as credit-based and start-stop (or on-off) signaling.[1] In the EVC design this information is communicated between routers that are k hops away, where k can take any value from 2 to l_{max}. This is because flits on k-hop EVCs that bypass $(k - 1)$ intermediate routers should be guaranteed a buffer at the kth router.

Limitations of buffer allocation

Kumar et al. use start-stop signaling to reserve the downstream buffers for the EVC flits.[2] Each router maintains a pool of shared buffers at each port that can be allocated to the EVC flits. When the number of free buffers falls below a calculated threshold Thr_k, the downstream node sends a *stop* signal to its upstream node k hops away. When the upstream node receives this signal, it stops sending flits. Similarly, when the number of free buffers at the downstream node exceeds the threshold Thr_k, the downstream node sends a *start* signal to the upstream nodes, so they can send more flits. This threshold value is computed based on the number of cycles it takes for the start-stop signals to propagate k-hops, taking into account all flits that could potentially arrive during this time (*buffer turnaround time*).

There are two problems with this approach. First, it limits EVCs' maximum length (l_{max}). To account for all possible flits that could arrive before the upstream router receives the *stop* signal, the minimum number of buffers required at each router grows as EVC length increases. Second, these threshold values accommodate the worst case—that is, the maximum number of flits in flight from all the nodes that would need downstream buffers. Typically, this overprovisioning of buffers would result in buffer wastage and underutilization of longer EVCs due to the high threshold of free buffers required for their operation. Thus, not only does this approach require more buffers, but the upstream nodes might find that they can't send EVC flits to the downstream node, despite the latter having free buffers. Longer EVCs also require greater wiring overhead for the reverse signals.[2]

Limitations of virtual channel allocation

In the original EVC design, head flits at upstream nodes can only arbitrate for a fixed number of output virtual channels of each type (1-hop, 2-hop, . . ., l_{max}-hop EVC). Suppose there are n nodes per dimension, and each router has v virtual channels per port. A static partition divides the v virtual channels into l_{max} bins, one bin for each type. The virtual channels in bin k refer to the input virtual channels at the downstream router that is k hops away. A dynamic partition of virtual channels isn't possible because it would require additional signaling overhead in the virtual channel allocation stage. For instance, if all upstream routers can reserve any input virtual channel at a downstream router, multiple routers might reserve the same virtual channel.

Global arbitration could avoid this situation, but it would require multiple cycles for these arbitration signals to move hop by hop from one router to another. Thus, each upstream node can arbitrate for only a fixed subset of a downstream router's virtual channels (per port) depending on the desired EVC type. The problem with this approach is that an upstream node can only send a fixed number of packets over EVCs to a particular downstream node k hops away. To send more packets, it must wait for a free EVC or use a shorter EVC, even though the downstream node might have other free virtual channels. As EVCs grow longer, this becomes a problem, because the number of virtual channels per bin decreases as l_{max} increases, allowing only one or two EVCs per downstream node. Moreover, longer EVCs mean that the signal indicating a free virtual channel would itself take multiple cycles to travel from the downstream to the upstream nodes. Thus even if a k-hop EVC becomes free, the upstream router would learn this only after k cycles, wasting a potential opportunity to use this EVC.

References

1. W.J. Dally and B. Towles, *Principles and Practices of Interconnection Networks,* Morgan Kaufmann, 2004.
2. A. Kumar et al., ''Toward Ideal On-Chip Communication Using Express Virtual Channels,'' *IEEE Micro,* vol. 28, no. 1, 2008, pp. 80-90.

point-to-point interconnects. Kumar et al. demonstrated the EVC flow-control optimization's impact but point out two deficiencies of the approach, both related to the many-cycle latency required for signaling and exchanging control information with the traditional short, dense, full-swing link design, coupled with the delivery guarantees expected of the NoC (see also the ''Express virtual channels'' sidebar).[1] First, buffers at

an EVC's end point must be managed conservatively to ensure that the destination router (the EVC end point) can accept the traffic. This leads to buffer overprovisioning and underutilization, which adversely impacts the network power dissipation. In fact, multiple researchers have shown that the power associated with NoC buffers accounts for 30 to 40 percent of the total power required for the NoC.[2,3] The second deficiency is that virtual channels must be partitioned between different express paths statically, and the control latency limits this number. As a result, the EVC design limits the number of nodes that an EVC flit can bypass to three or fewer.

Using our Nochi approach and proposed interconnect circuits, we enable single-cycle control communication across all nodes in a row or column of a mesh network, alleviating EVC limitations. Using timely information reduces the demand on router buffers as well as the number of buffers needed to sustain a specific bandwidth, and hence reduces the critical buffer leakage power. At the same time, allowing distant nodes to instantaneously claim EVCs increases the EVC technique's applicability, enabling more bypassing of nodes and reducing the traversal latency and dynamic power required to deliver packets across large distances on the chip.

Global interconnect circuits

In this work, we extend capacitive feed-forward circuits with multipoint broadcast ability and collision detection with node quantity determination. This provides the capability for global chip communication with single-cycle latency. Capacitive links offer a modest to good improvement over the best achievable latencies using repeated resistance-capacitance links, but with significant power reduction. Although latency isn't as low as transmission-line or optical techniques (see the "Related work in advanced interconnects" sidebar), capacitively driven links offer the most desirable trade-off between power and area while still achieving single-cycle latency.

For our simulations, we use a 65-nm, $V_{dd} = 1$ V standard CMOS process with eight metal layers. Figure 1 shows a block diagram of one column of the 7×7 chip multiprocessor we evaluate for the sample Nochi design point. Assuming a 7-millimeter chip edge, top-level thick M8 metal, and a low-κ dielectric, a 1-mm wire of width 1 μm has a lumped resistance of 20Ω with a total coplanar capacitance of approximately 400 fF. Given these dimensions, we sized the feed-forward capacitor at 300 fF, consuming 5×5 μm^2 implemented using metal-oxide-semiconductor (MOS) transistors and requiring a medium-sized inverter buffer to drive it. Because the feed-forward capacitor decouples the common mode voltage of the driver from the wire, large 30kΩ termination resistors are used to set the differential wire's common-mode voltage.

Figure 2 shows the pulse response at 1-mm locations from one end of the global line to the other. Notice that the feed-forward capacitance not only increases bandwidth, but also provides a pre-emphasis capability that helps compensate for high-frequency signal attenuation. The delay from core 0 to core 6 is 192 picoseconds, or within a single clock cycle in our design. (Note that we use the terms *core, router,* and *node* interchangeably.)

Our proposed circuit design provides global multidrop capability with single-cycle latency and lets the receiver sense the number of transmitters using the line on a per-bit granularity. We refer to this technique as *smart carrier sense multiple access.* To perform S-CSMA, we implement a flash, analog-to-digital converter (ADC) that implements voltage amplitude sensing to determine the number of transmitters at any one instance. We refer to the case when there are p transmissions on the line as a p-level S-CSMA. The worst-case situation for multiple transmitters colliding with each other is the longest shared, multidrop bus, in which six cores simultaneously communicate with the seventh, and farthest, core ($p = 6$). In this situation, six voltage levels are possible, requiring a six-level ADC running at 2.5 GHz to determine the number of simultaneous transmitting cores.

Figure 3 shows the eye diagram over 2K cycles of the six voltage levels. The minimum eye opening is approximately 79 mV, which is large enough to overcome any quantizer offset and input sensitivity limitations. The current

Related work in advanced interconnects

Flow control, the mechanism that allocates resources to packets within the network, is a key determinant of communication energy/delay and network throughput. As a result, a vast body of work in this area exists. We focus on approaches that leverage advanced interconnect circuits. Note that the baseline router we use for comparison incorporates many recently proposed techniques for improving network energy/delay, such as speculation,[1] bypassing,[2] look-ahead pipelines,[3] simplified virtual-channel allocation,[3] and look-ahead routing.[4] These techniques drive toward ultralow router latency, but only succeed in bypassing the pipeline at low loads, performing poorly when network contention is high.

Kim and Stojanovic use equalized interconnects in on-chip network designs.[5] However, they focus on existing on-chip network topologies, whereas we extend and modify flow control to leverage low-latency characteristics. Another approach, flit-reservation flow control, harnesses the faster upper metal wires to send control flits out in advance to schedule resources for subsequent data flits, allowing data flits to zoom through the pipeline when they arrive.[6] However, this technique needs large reservation tables and a complex router microarchitecture.

More disruptive interconnect technologies, such as on-chip photonics[7] and RF interconnects,[8] enable high-bandwidth global communications, which could mandate a rethinking of on-chip network designs as these technologies develop. Our G-line interconnect is a nearer-term technology that can be readily fabricated in today's VLSI technologies.

Several proposals show the potential for communicating at light speed across several millimeters on a silicon substrate. Some researchers show early point-to-point circuits that allow transmission-line, wavelike velocity for 10 mm of interconnect.[9] Although transmission-line circuits achieve a speed of 10 picoseconds per millimeter in silicon dioxide, they require a large area per differential pair and high power consumption. Even worse, because they use current-mode signaling, they consume significant static power even with little activity.

Other work proposes an RF interconnect for networks on chip.[8] RF-I provides a shortcut path between distant nodes via a single RF transmission line. It achieves up to 30 gigabits per second on a single wire in 90-nm CMOS using frequency-division multiple access with an energy consumption of 1.2 pJ per bit. However, at 30 Gbps, an RF-I bandwidth density of 2.5 Gbps/μm is on par with much simpler interconnect buses that maintain smaller transceiver footprints, lower power consumption, and easier design integration.

Optical interconnects offer obvious latency benefits across long distances and use wave division multiplexing to allow multiple channels of data across a single physical interface.[10]

Recent work by Ito et al. allows both low latency and multidrop ability on a transmission line with low-power dissipation of 1.2 mW per transceiver.[11] They suggest that broadcast, multidrop, bidirectional ability is achievable for NoC applications; however, they don't show how to arbitrate or schedule such information. Others have shown that a capacitive feed-forward method of global interconnect achieves nearly single-cycle delay for long RC wires, but with voltage-mode signaling.[12] By using a simple inverter driving a feed-forward capacitance, this technique exchanges voltage gain for bandwidth. Our G-line circuits extend this technique through multidrop connectivity and smart carrier sense multiple access (S-CSMA) collision-detection and measurement techniques.

References

1. L.-S. Peh and W.J. Dally, ''A Delay Model and Speculative Architecture for Pipe-lined Routers,'' *Proc. Int'l Symp. High Performance Computer Architecture* (HPCA 01), IEEE CS Press, 2001, pp. 255-266.
2. S.S. Mukherjee et al., ''The Alpha 21364 Network Architecture,'' *IEEE Micro,* vol. 22, no. 1, Jan./Feb. 2002, pp. 26-35.
3. A. Kumar et al., ''A 4.6Tbits/s 3.6GHz Single-Cycle NoC Router with a Novel Switch Allocator in 65nm CMOS,'' *Proc. Int'l Conf. Computer Design* (ICCD 07), IEEE CS Press, 2007, pp. 63-70.
4. M. Galles, ''Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI Spider Chip,'' *Proc. Hot Interconnects 4* (HOTI 96), IEEE CS Press, 1996, pp. 141-146.
5. B. Kim and V. Stojanovic, ''Equalized Interconnects for On-Chip Networks: Modeling and Optimization Framework,'' *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 07), IEEE Press, 2007, pp. 552-559.
6. L.-S. Peh and W.J. Dally, ''Flit-Reservation Flow Control,'' *Proc. Int'l Symp. High Performance Computer Architecture* (HPCA 00), IEEE CS Press, 2000, pp. 73-84.
7. A. Shacham, K. Bergman, and L.P. Carloni, ''The Case for Low-Power Photonic Networks on Chip,'' *Proc. Design Automation Conf.* (DAC 07), ACM Press, 2007, pp. 132-135.
8. M.F. Chang et al., ''CMP Network-on-Chip Overlaid with Multi-Band RF-Interconnect,'' *Proc. Int'l Conf. High-Performance Computer Architecture* (HPCA 08), IEEE CS Press, 2008, pp. 191-202.
9. A. Jose, G. Patounakis, and K.L. Shepard, ''Pulse Current Mode Signalling for Nearly Speed-of-light Intrachip Communications,'' *IEEE J. Solid State Circuits,* vol. 41, no. 4, Apr. 2006, pp. 772-780.
10. S. Palermo, A. Emami-Neyestanak, and M. Horowitz, ''A 90-nm CMOS 16 Gb/s Transceiver for Optical Interconnects,'' *IEEE J. Solid-State Circuits,* vol. 43, no. 5, May 2008, pp. 1235-1246.
11. H. Ito et al., ''A Bidirectional- and Multi-Drop Transmission-Line Interconnect for Multipoint-to-Multipoint On-Chip Communications'' *IEEE J. Solid-State Circuits,* vol. 43, no. 4, Apr. 2008, pp. 1020-1029.
12. E. Mensink et al., ''A 0.28pf/b 2Gb/s/ch Transceiver in 90-nm CMOS for 10-mm On-Chip Interconnects,'' *Proc. IEEE Solid-State Circuits Conf.* (ISSCC 07), IEEE Press, 2007, pp. 412-413.

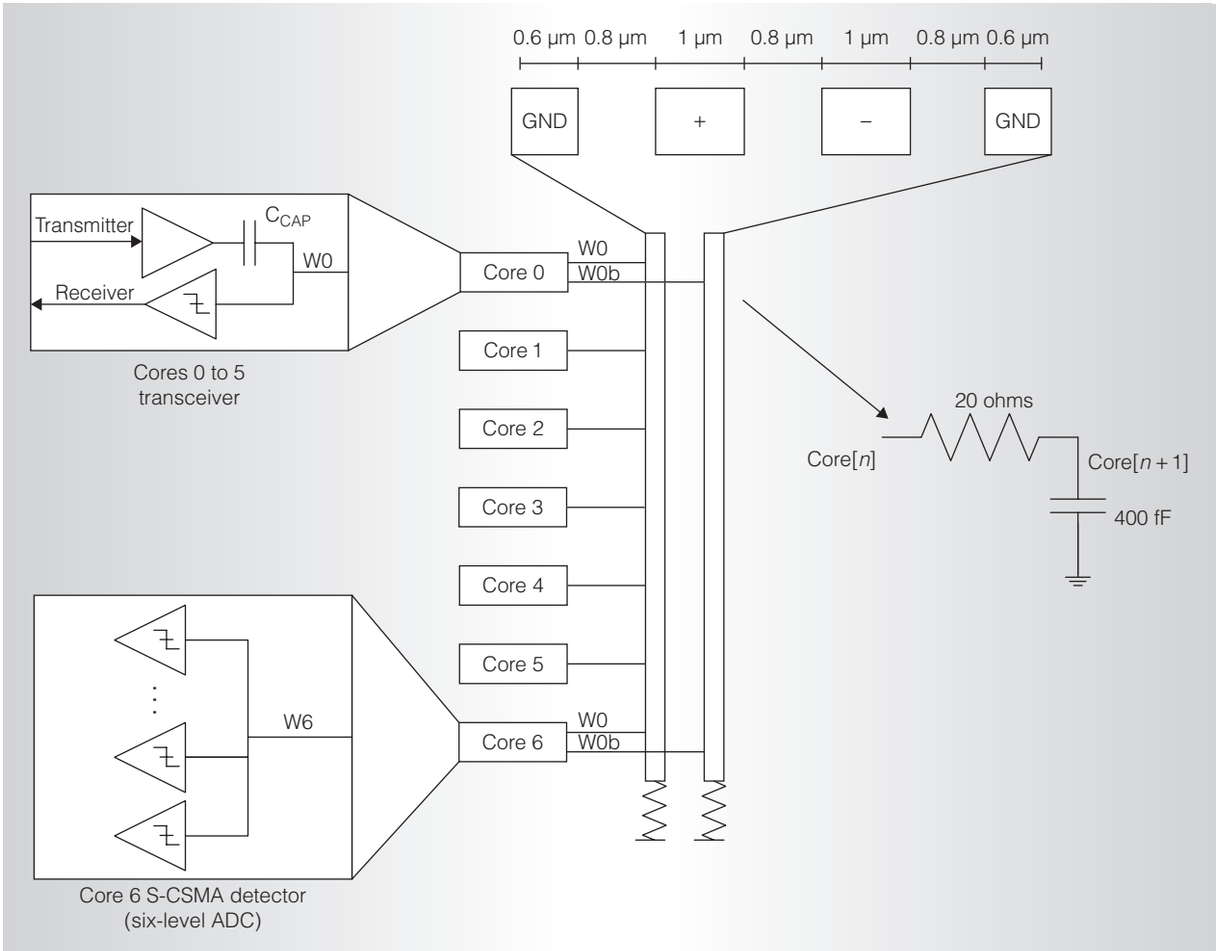

Figure 1. Block diagram and schematic of a seven-core global interconnect. Transceivers are shown as single ended for clarity.

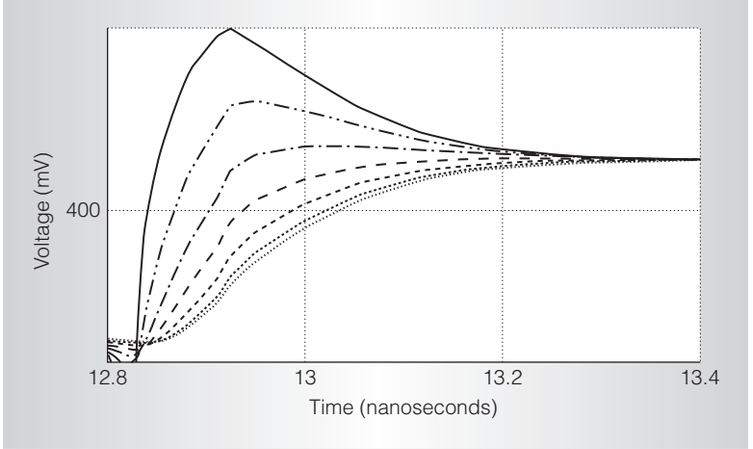

Figure 2. Step response and pulse propagation at all seven core locations. The delay from core 0 to core 6 is 192 ps.

design uses receiver offset cancellation[4] to improve the minimum eye opening sensitivity to approximately 20 mV.

The simulated power dissipation is 0.6 mW per transmitter and 0.4 mW per receiver-quantizer (a single receiver uses 0.4 mW, whereas a six-level S-CSMA receiver uses 2.4 mW).

Flow control for EVCs using global interconnects

The flow control scheme for networks involves managing the buffers and virtual channels at each router for efficient utilization. Flits can proceed from one router to the other only after acquiring buffers and virtual channels (because dropping of flits isn't allowed). In the EVC design, the allocation

of virtual channels determines the type of EVCs (1-hop, 2-hop, and so on) that flits acquire, which they use to bypass intermediate routers. Routers exchange the buffer flow control information using start-stop signaling in the original EVC design. A *start* token from a downstream router indicates that it has free buffers and sending is allowed, whereas a *stop* signal indicates that sending isn't allowed. To scale EVC lengths and remove the original EVC design's limitations (see the "Express virtual channels" sidebar), we need

- a low-latency communication fabric for faster propagation of the buffer flow-control bits between routers, which would help reduce buffer turnaround time; and
- global arbitration of the virtual channels rather than static local assignment, which would help better allocate network resources based on traffic.

Using global metal links for sending flow-control information and a scheme for arbitration over these wires could help achieve these goals. However, full-swing repeated RC global interconnects consume a lot of power, and sending flow-control information over these long links, potentially every cycle, might make it difficult for the network to meet the power budget. On the other hand, using our capacitively driven, low-swing interconnects would give us a win-win situation: a communication fabric that is both low latency and low power.

In our flow-control mechanism for EVCs, the capacitively driven G-lines broadcast control signals along a dimension. This can help overcome the EVC length limitations, potentially reducing power and improving performance.

Nochi EVC design

G-lines provide a single-cycle broadcast across the chip. Our design uses G-lines for sending start-stop signals. A node could therefore send a *start* signal upstream to the point at which only one empty buffer is left, enabling EVCs of arbitrary lengths. For an $n \times n$ chip, EVCs can have maximum lengths of up to $n - 1$ (the maximum possible hops per direction). A flit can thus bypass all routers along its path in one direction, get buffered at the last router in its path in that direction, turn, and zoom all the way until it reaches its destination node. (We assume *x-y* routing, and that EVCs can't turn, as in the original EVC design.)

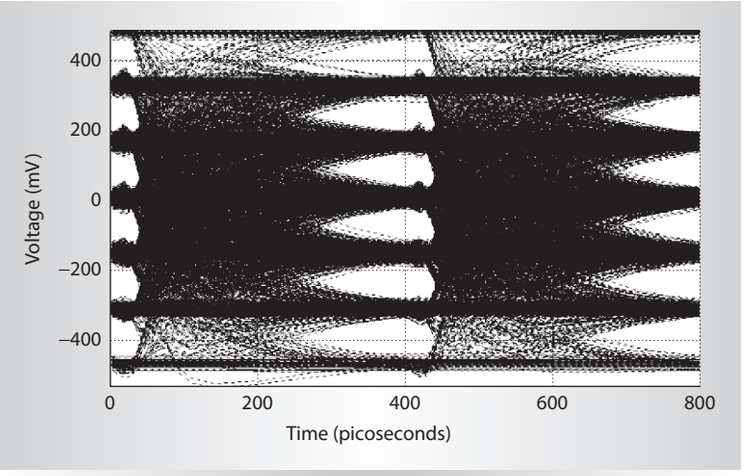

Figure 3. Six-level eye diagram determining the number of simultaneous transmissions based on voltage.

We can also use G-lines to signal the availability of free virtual channels at each node. This lets nodes dynamically arbitrate for all EVCs at a particular router port, instead of a few statically assigned ones. An upstream node can then select multiple virtual channels to the same downstream node, allowing flits to travel on longer EVCs as far as possible, and thereby reducing latency.

For our flow-control mechanism we chose a 49-core chip multiprocessor (CMP) with a 7×7 packet-switched mesh network. We allow up to six-hop EVCs ($l_{max} = 6$), meaning flits can potentially bypass all routers in a direction. Figure 4 compares our Nochi EVC design with a state-of-the-art router pipeline design (baseline) and the original EVC design.[1]

We assign 14 one-bit G-lines per direction (N-S, S-N, E-W, and W-E), as Figure 5 shows. We divide these G-lines into two sets: one for virtual channel signaling and one for buffer signaling. Each G-line is statically assigned to one core. The rules for signaling are:

- A downstream router transmits on its assigned G-line if it wants to signal to

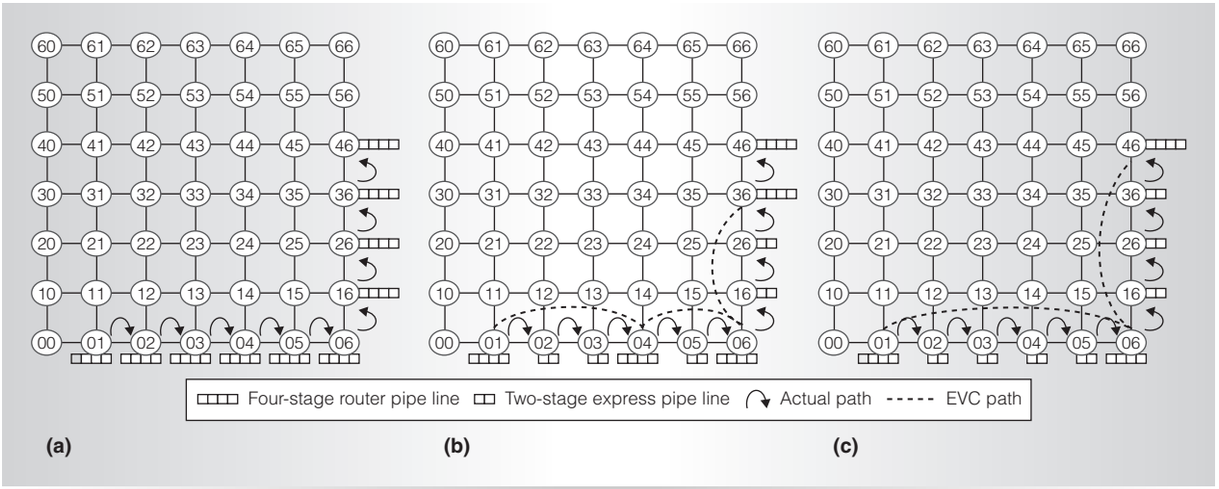

Figure 4. Comparison of baseline (a), original EVC (b), and Nochi EVC (c) designs for a flit travelling from node 01 to node 46. In the baseline, the flit goes through the four-stage router pipeline at every hop. In the original EVC design, the flit takes a three-hop EVC to node 04, a two-hop EVC to node 06, a three-hop EVC to 36, and finally a one-hop NVC to node 46. In the Nochi EVC design, the flit takes a five-hop EVC to node 06 (the point of turning), and a four-hop EVC to node 46.

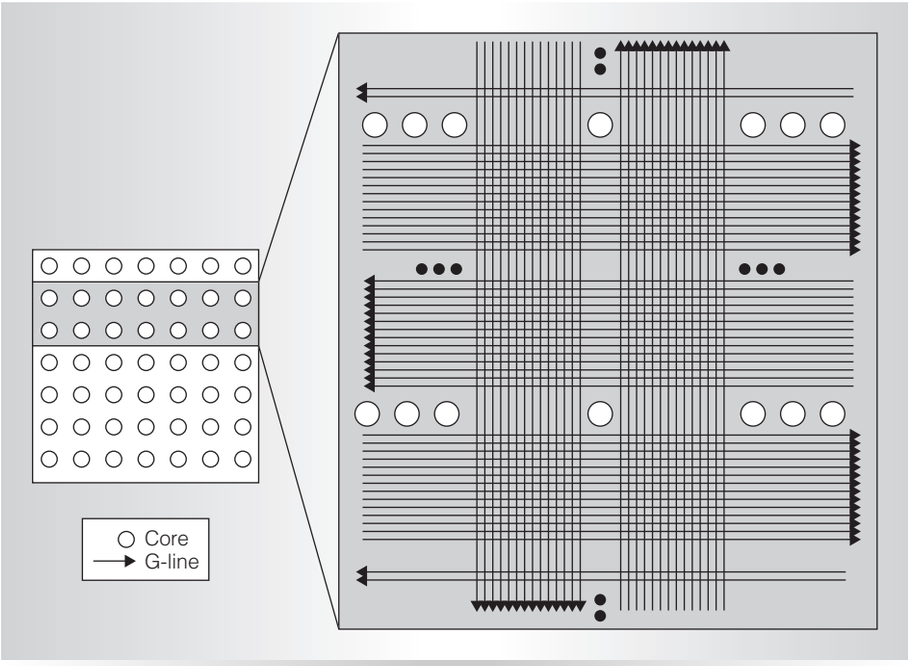

Figure 5. A 7 × 7 chip with 14 G-lines per direction per row and column. A router can only receive flits from its upstream routers along a particular direction.

all its upstream routers that it has free buffers or virtual channels.
- All upstream routers can signal to a particular downstream router by transmitting on its assigned G-line.
- The downstream and upstream routers signal alternately on the G-lines.

Because there are two sets of G-lines, each node has one G-line it uses to broadcast

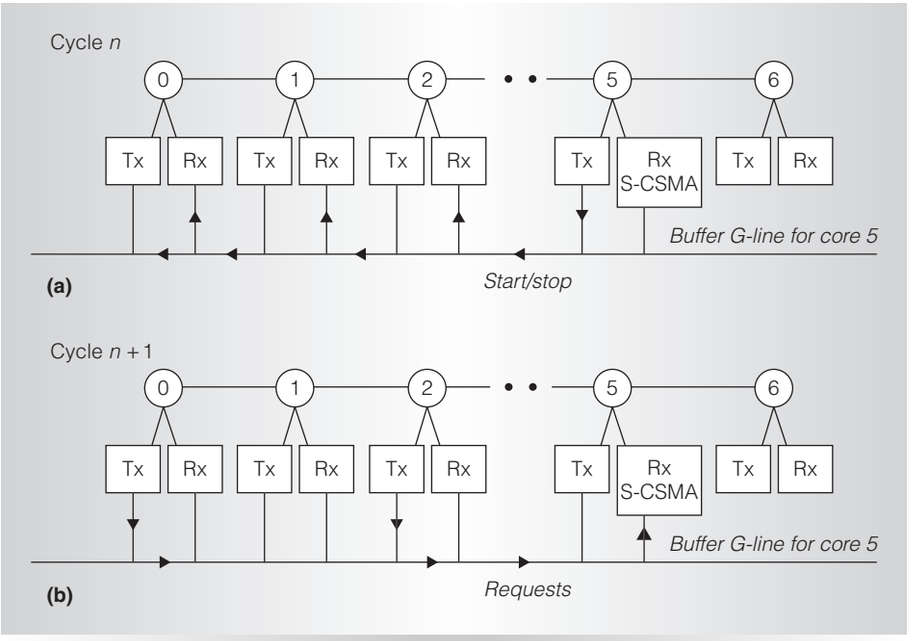

Figure 6. Buffer signaling over G-lines for router 5 (Rx: receiver; Tx: transmitter). In cycle *n*, router 5 transmits a 1 on its buffer G-line if it has a free buffer, and all its upstream nodes (router 0 to router 4) snoop the G-line (a). In cycle $n + 1$, router 0 and router 2 place requests for router 5's buffers by transmitting on this G-line; router 5 receives and performs S-CSMA to calculate the number of requests (b).

information about its free virtual channels and another to indicate its free buffers. Upstream routers can send flits to the node and arbitrate for free virtual channels and buffers over the node's G-lines. Each downstream router has a special receiver that counts the number of signals transmitted on the G-line that cycle. The downstream router can use this S-CSMA property of the G-line receivers to calculate how many upstream nodes are requesting its virtual channels or buffers, and grant requests accordingly. The downstream router doesn't need to know which particular upstream nodes placed the requests; it just needs to know the number of requests. The downstream router reserves the requested number of buffers and virtual channels for the flits that will eventually arrive. (Note that upstream nodes reserve downstream buffers before transmitting flits because dropping flits isn't permitted.)

Virtual channel and buffer signaling

Figure 6 illustrates buffer signaling for a downstream router (router 5) without loss of generality. Virtual channel signaling occurs the same way. All of router 5's upstream routers in the W-E direction—that is, routers 0 to 4—can send flits to it via the 5- to 1-hop EVCs, respectively. The downstream node and the upstream nodes can transmit on the G-line every alternate cycle. In cycle *n*, router 5 transmits a 1 (a *start* signal) on its G-line if it has at least one free buffer. All downstream routers snoop on this G-line and learn that router 5 has a free buffer. In cycle $n + 1$, the upstream nodes that want to reserve a buffer at this node for their flits transmit a 1 on the G-line. Router 5's receiver performs S-CSMA to calculate how many routers want to send it flits, decreasing its free buffer count accordingly (implying the reservation of buffers for the flits that will arrive from the requesting upstream nodes).

If there are more buffer requests to the downstream node than there are available buffers, the number of requests granted will equal the number of free buffers. The free buffer count becomes zero, so router 5

doesn't transmit a 1 on its buffer G-line in the next cycle.

Next, router 5 transmits the count of the number of requests it granted to all its upstream nodes via normal wires. Each upstream router that is waiting for an acknowledgement of its buffer request to router 5 can determine whether its request was granted based on the value of the count it receives and some priority scheme (in case the number of requests exceeds the number of grants). In our design, the upstream node that receives the count first checks whether it issued a buffer request to router 5, and if it did, decrements this count and forwards the new count upstream. The node also signals to its switch allocator that the buffer request was granted and the flit can proceed. As the grant count propagates upstream, the requesting nodes continue decrementing the value. Thus, a node receiving a count of 0 knows that its request wasn't granted and places a new request the next time router 5 transmits a 1 or places a new request on another router's G-line. Router 5, meanwhile, continues updating its free buffer count as flits leave.

When there are enough buffers and virtual channels, router 5 transmits a 1 on its G-lines every other cycle while the grant count propagating on normal wires allocates the buffers/VCs to the requesting upstream node. Thus, multiple requests for buffers and EVCs at router 5 could be granted every other cycle.

Transmitter and receiver operation

Each downstream node has a different number of upstream nodes. For instance, the upstream nodes in the W-E direction for router 6 are routers 0 to 5; for router 5, they're routers 0 to 4; for router 4, they're routers 0 to 3; and so on. Thus, the total number of G-line transmitters per direction is $6 + 5 + 4 + 3 + 2 + 1 = 21$ for each G-line type (buffer and virtual channel signaling), making 42 the total number of G-line transmitters per direction. However, these aren't all active every cycle. During the cycle when downstream nodes transmit buffer and virtual channel availability on their respective G-lines, the upstream nodes aren't allowed to transmit. Hence, there can be a maximum of 12 transmissions per row or column per direction (if all downstream nodes have free buffers and free virtual channels). During the cycle when upstream nodes transmit buffer and virtual channel requests, they can't send buffer and virtual channel requests to multiple downstream nodes in the same cycle. Thus, there can be at most 12 transmissions per row or column per direction (if all upstream nodes transmit buffer and virtual channel requests on some G-line). Thus, in every cycle, there can be at most 12 active transmitters per row or column per direction. The total number of active transmitters across the whole network would thus be $12 \times 2 \times 2 \times 7 = 336$ out of a total of $42 \times 2 \times 2 \times 7 = 1{,}176$ transmitters. These transmissions might be the downstream nodes signaling buffer and virtual channel availability or upstream nodes transmitting requests in alternate cycles. The percentage of active transmitters in every cycle is $336/1{,}176 = 28.5$ percent.

Similarly, there are 42 G-line receivers per row per direction. However, the upstream receivers snoop on all the G-lines every alternate cycle looking for buffer and virtual channel availability. They're all thus active every alternate cycle, making 1,176 active receivers in the network. During request signaling by upstream nodes (every alternate cycle), only the downstream routers have active receivers. Thus, there are 12 active receivers in this cycle. The total number of active receivers in the whole network in these cycles is 336. However, these receivers are the S-CSMA receivers, which are more complex than normal upstream receivers. The receiver for router 6 must perform a six-level S-CSMA (as routers 0 to 5 can all be transmitting to router 6 on its G-line), while router 5's receiver must perform a five-level S-CSMA, and so on. We've considered these estimates of the maximum possible transmitters and receivers active every cycle when calculating the G-line transmitter and receiver power.

Buffer signaling optimization

An extra optimization is that for EVCs of length 3 or less, in addition to the G-line-based signaling, we use traditional

threshold-based start-stop signaling over local wires. This optimization decreases starvation of neighboring nodes due to long EVCs. This signaling is based on buffer thresholds, whereas the G-line signaling continues until no buffer is available. This hybrid flow-control scheme using G-lines in conjunction with normal wires ensures that this new implementation will achieve at least the original EVC performance.

Evaluation

We performed limit studies to evaluate the Nochi-based flow control's potential for EVCs using both synthetic and real traffic on packet-switched mesh topologies. We also studied the effect of using the G-lines on a ring topology. Table 1 presents the microarchitecture and process parameters, which remain the same across all the studies.

Table 1. Process and network parameters.

Parameter	Measure
Technology parameters	
Technology	65 nm
V_{dd}	1 V
Frequency	2.5 GHz
Network parameters	
Routing	*XY* dimension-ordered
Number of router ports	5
Virtual channels per port	8
Flit size/channel width (c_{width})	128 bits
Link length	1 mm
Wire pitch (W_{pitch})	0.45 μm
Global interconnect circuit parameters	
Differential pair width	5.6 μm
Capacitive-feed-forward inverter power (transmitter)	0.6 mW
Offset-canceled quantizer power (receiver)	0.4 mW
7-mm wire pulse-propagation latency	192 ps

Design assumptions

For our evaluations, we made certain assumptions about the flow-control design to estimate our scheme's potential. For buffer signaling, we assume that the downstream node's free-buffer signaling and the upstream nodes' arbitration for these buffers occur within a cycle.

When there is contention for buffers (that is, when there are more requests than free buffers) or VCs, a static priority exists: the requesting node farthest from the downstream node has the highest priority for getting a buffer/VC, then the next farthest, and so on. The situation for virtual channel allocation is similar.

Synthetic traffic

Using synthetic traffic, we compared Nochi EVC with a conventional EVC design.[1] We evaluated the network performance using an industry in-house, cycle-accurate simulator that models all the router pipeline's major components at clock granularity.

We calculated router power based on extrapolations from Hoskote et al.[2] for our design point. In our results, saturation throughput is the point at which packet latency becomes three times the no-load latency. To be consistent, we also used a 7×7 mesh topology here. We used both uniform random traffic (in which each node sends packets to randomly chosen destinations) and tornado traffic (in which each node sends packets halfway around the mesh along the *x*-dimension) to evaluate Nochi EVC. The lower average hop count along each dimension in uniform random traffic led to less utilization of long express paths, resulting in almost similar performance for Nochi EVC and baseline EVC.

For tornado traffic, in which packets must travel more hops along a dimension, the use of longer express paths is high, leading to a significant performance gain. Figure 7a plots flit latency as a function of network load for tornado traffic for both Nochi EVC and baseline EVC, assuming the same amount of buffering (25 buffers per port). The lower latency with Nochi EVC is mainly due to longer express paths. These paths let packets bypass on average 53.7 percent of the routers along their path, as compared to 41.3 percent with original EVC.

To achieve a given saturation throughput, Nochi EVC requires significantly fewer buffers than original EVC. Specifically, a Nochi design with 15 buffers per port exhibits the same saturation throughput as the original EVC design with 25 buffers per port, as Figure 7b shows. To evaluate the resulting reduction in buffer power, we used extrapolated data based on the Intel Teraflops NoC router.[2] We scaled the router

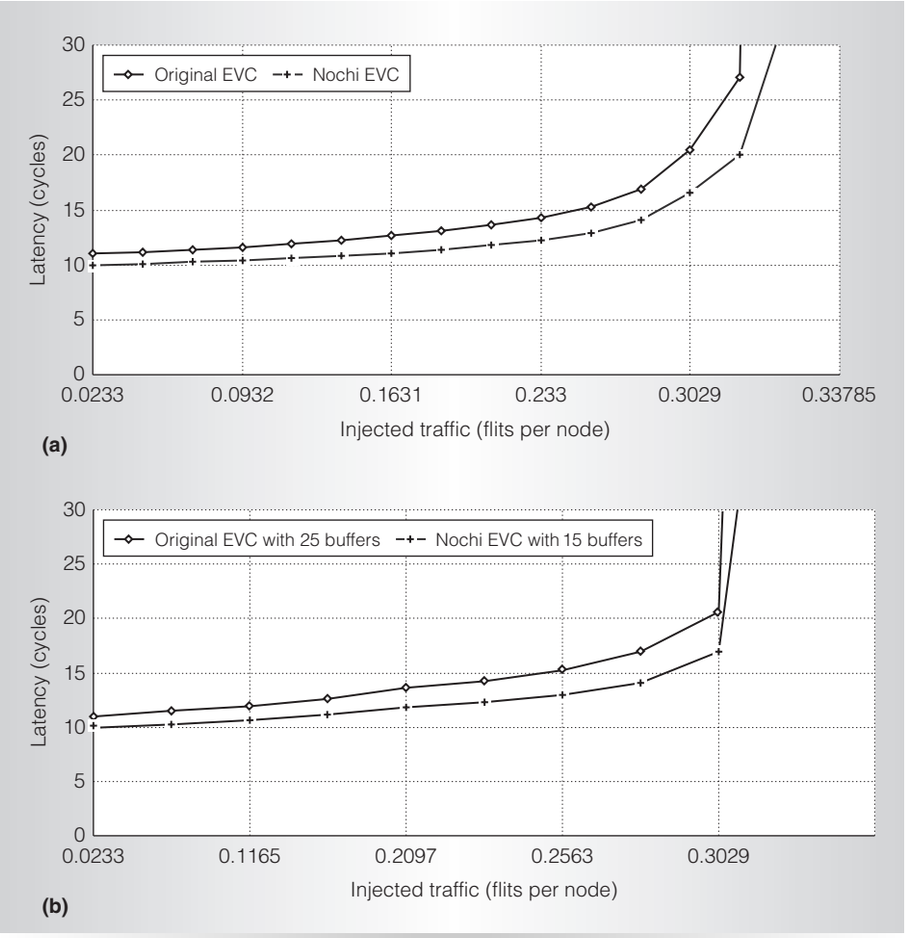

Figure 7. Network latency with tornado traffic for a 7 × 7 mesh (a). Nochi EVC reduces latency by 44 percent near the EVC saturation point and 9.4 percent at no-load. Nochi EVC achieves the same saturation throughput as baseline EVC with fewer buffers (b).

power of 924 mW at 5 GHz down to 2.5 GHz at 1.0, deriving scaled router power of 500 mW. The Teraflops router has sixteen 38-bit-wide buffers per lane and two lanes per port, for a total of 6,080 buffer bit cells per router. The buffers consumed 22 percent of this power. For the 128-bit-wide lanes in our design, the power per buffer thus becomes 11.6 mW. We assume that 60 percent of this power is dynamic and 40 percent is leakage. Because Nochi needs 10 fewer buffers per port than baseline EVC and packets bypass 12.4 percent more nodes on average, we get a dynamic buffer power reduction of 53.7 mW per router and a buffer leakage power reduction of 46.3 mW per router.

Thus, the reduction in buffer power for Nochi EVC is 100 mW per router (4.9 W for the entire 49-node network), or 45.9 percent less than the conventional EVC design. However, our circuit-level simulations show that G-lines consume a total of 0.67 W (as we describe next). Hence, the net power reduction of Nochi EVC compared to the original EVC is 4.23 W, or 8.8 percent of the total network power (extrapolated from the Teraflops data by assuming 128-bit buffers, crossbars, and links as compared to 38 bit).

G-line power calculation

Our initial circuit design, which we later abandoned, used 50 transmission lines on die, with 10-µm differential wire pitch

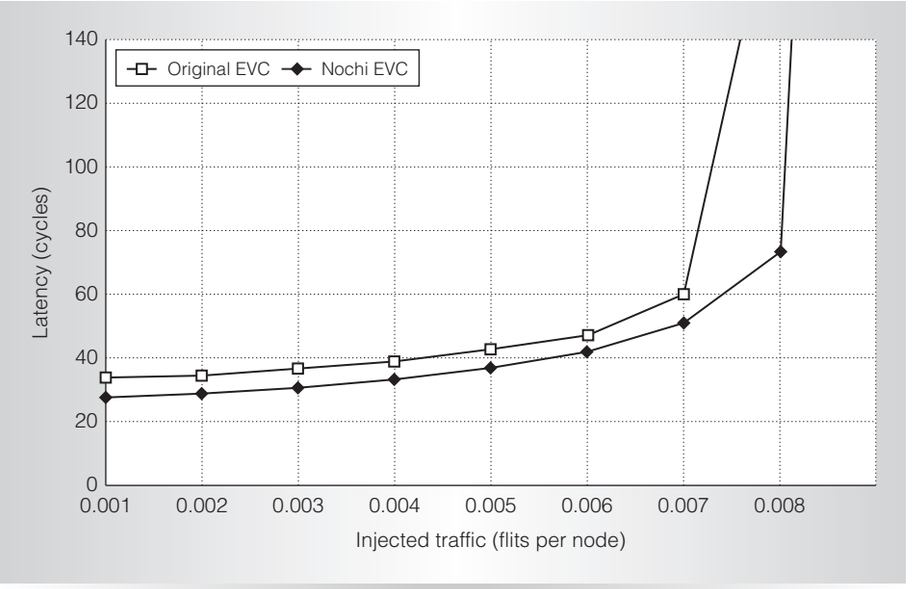

Figure 8. Network latency with tornado traffic for a 16-node ring. The longer EVC paths in the Nochi EVC design result in a 17 percent reduction in no-load latency and a 58 percent reduction in latency near saturation.

(2.5-µm width and 2.5-µm spacing). Although the transmission line exhibited a near-speed-of-light 10-ps/mm phase velocity, the series DC wire resistance presented a critical problem. For a wire 7 mm long, the series resistance was on the order of the characteristic impedance, resulting in significant signal attenuation. This series resistance, coupled with the low impedance attained on die, resulted in current dissipation of 5 mA per transmitter. In addition, the series resistance made it difficult to distinguish the exact number of transmitters that were simultaneously sending signals on the G-lines to enable S-CSMA.

Fortunately, the proposed forward-C capacitive transceiver exhibits little power because the series capacitor reduces the global interconnect swing by a factor of 8 to 10. In addition, no static power is consumed using the G-line circuit. With a simulated 0.6 mW per transmitter, and a total of 336 transmitters operating at once, the total chip transmit power is 0.2 W. The receiver power consists of the switching power needed to quantize a small differential input voltage. The speed and power of such quantizers scale well with CMOS process scaling, such that in our 65-nm CMOS process, quantization power (including clock power) is 0.4 mW per receiver. This quantizer power is the same for a 1-bit receive and a one-level decision within an S-CSMA analog-to-digital converter.

A p-level S-CSMA receiver's power is thus p times that of a single quantizer. The total of 336 receivers operating in a given cycle is equivalent to 1,176 quantizers, making the total chip receiver power 0.47 W. So, the entire power consumed by the transmitters and receivers is 0.67 W.

Sensitivity to topology

We observed the impact of using G-lines for fast transmission of control messages over a ring network, such as those used in commercial chips (for example, Cell Processor and Larrabee). Experiments on an eight-node ring didn't yield significant differences between the original EVC and Nochi EVC designs (for both uniform random and tornado traffic), because the maximum number of hops in either direction is only four, and hence the longer EVC paths enabled by the G-lines aren't needed. However, with a 16-node ring and tornado traffic, we see a significant reduction in latency due to Nochi's longer EVC paths, as Figure 8 illustrates.

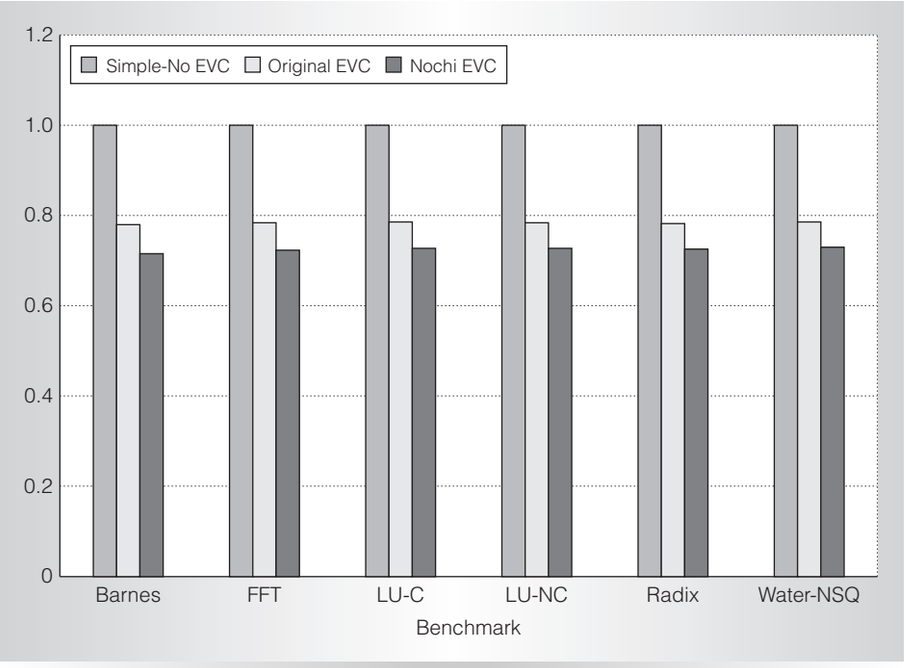

Figure 9. Normalized network latency using Splash-2 traffic for an 8 × 8 mesh comparing Nochi-EVC with the original EVC design and a baseline network design.

Nochi with real traffic

We also studied the Nochi EVC scheme using real network traffic. We performed full-system simulations using Virtutech Simics extended with the GEMS tool set.[5] We extended the Garnet network model,[6] which models the router pipeline including virtual channels, buffers, allocators, and switches, to capture the aspects of the G-line EVC protocol.

We ran Splash-2 benchmarks on a 64-core CMP with shared level-two (L2) caches distributed in a tiled manner. Each core consists of a two-issue in-order Sparc processor with 64 Kbytes level-one instruction and data caches. Each tile also includes a 1-Mbyte L2 bank. We attached DRAM to the CMP via eight memory controllers along the edges. We used an 8 × 8 mesh for the on-chip network and the MOESI-based directory protocol for cache coherence. We evaluated three network configurations: a baseline network with no EVCs, a network with the original EVCs, and a network with Nochi EVCs.

Figure 9 plots the observed normalized network latency. On average, the Nochi EVC scheme provides a 27.5 percent improvement over a simple network and a 7.6 percent improvement over the original EVC design. In addition, the average percentage of routers bypassed increases from 46.7 percent for baseline EVC to 59.1 percent for the Nochi EVC design. The Splash-2 benchmarks don't stress the network much, making the network operate at similar design points across all benchmarks; hence, the network optimization scheme results in an almost constant network speedup.

In addition to the G-line EVC design described here, the Nochi approach is more general, and is applicable to any technology in which the underlying properties offer trade-offs between near-immediate signaling and bandwidth, such as optical or RF interconnect. We're currently evaluating other applications that could potentially benefit from near-instantaneous global information, such as synchronization, dynamic power management, cache coherence, and so on. MICRO

Acknowledgments

US National Science Foundation award 0811798 and the Marco Interconnect Focus Center supported this work.

References

1. A. Kumar et al., ''Express Virtual Channels: Towards the Ideal Interconnection Fabric,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 07), ACM Press, 2007, pp. 150-161.
2. Y. Hoskote et al., ''A 5-GHz Mesh Interconnect for a Teraflops Processor,'' *IEEE Micro,* vol. 27, no. 5, Sept./Oct. 2007, pp. 51-61.
3. H.-S. Wang, L.-S. Peh, and S. Malik, ''Power-Driven Design of Router Microarchitectures in On-Chip Networks,'' *Proc. Int'l Symp. Microarchitecture* (Micro 03), IEEE CS Press, 2003, pp. 105-116.
4. M.-J.E. Lee, W. Dally, and P. Chiang, ''Low-Power Area-Efficient High-Speed I/O Circuit Techniques,'' *IEEE J. Solid-State Circuits,* vol. 35, no. 11, Nov. 2000, pp. 1591-1591.
5. M.M.K. Martin et al., ''Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset,'' *SIGARCH Computer Architecture News,* vol. 33, no. 4, 2005, pp. 92-99.
6. N. Agarwal et al., ''Garnet: A Detailed On-Chip Network Model inside a Full-System Simulator,'' *Proc. Int'l Symp. Performance Analysis of Systems and Software* (ISPASS 09), IEEE CS Press, 2009, pp. 33-42.

Tushar Krishna is a PhD candidate in the Department of Electrical Engineering at Princeton University. His research interests include networks on chip and parallel computer architectures. Krishna has a bachelor's degree in electrical engineering from IIT Delhi, India. He is a student member of the IEEE and ACM SIGARCH.

Amit Kumar is a research scientist in Intel's Microarchitecture Research Lab in Santa Clara, California, and a PhD candidate in the Department of Electrical Engineering at Princeton University. His research interests include on-chip networks, computer architecture, and power- and thermal-aware system design. Kumar has an MA in electrical engineering from Princeton University. He is a member of the IEEE and ACM.

Li-Shiuan Peh is an associate professor of electrical engineering at Princeton University. Her research focuses on low-power interconnection networks, on-chip networks, and parallel computer architectures. Peh has a PhD in computer science from Stanford University.

Jacob Postman is a PhD student in electrical and computer engineering at Oregon State University, Corvallis. His research interests include low-power, high-performance circuits, microarchitectures, and networks on chip. Postman has a BS in electrical and computer engineering from Oregon State University, Corvallis.

Patrick Chiang is an assistant professor of electrical and computer engineering at Oregon State University. His interests are in the design of low-power, process-calibrating, mixed-signal circuits in deep submicron CMOS. Chiang has a PhD in electrical engineering from Stanford University.

Mattan Erez is an assistant professor of electrical and computer engineering at the University of Texas at Austin. His research interests include computer architecture and programming models. Erez has a PhD in electrical engineering from Stanford University.

Readers can contact Tushar Krishna at B-308, Dept. of Electrical Eng., EQUAD, Olden St., Princeton, NJ 08544; tkrishna@princeton.edu.