Toward Ideal On-Chip Communication Using Express Virtual Channels

Current on-chip networks use a packet-switched design with a complex router at every hop, which imposes significant communication energy, delay, and throughput overhead. We propose reducing energy and delay, and increasing throughput, using express virtual channels. Packets traveling along these virtual express lanes, which connect distant nodes in the network, bypass intermediate routers, significantly reducing router overhead.

Amit Kumar
Li-Shiuan Peh
Princeton University

Partha Kundu
Intel

Niraj K. Jha
Princeton University

Driven by technology limitations to wire scaling and increasing bandwidth demands, packet-switched on-chip networks are quickly replacing shared buses and dedicated wires as the de facto interconnection fabric in general-purpose chip multiprocessors (CMPs)[1,2] and application-specific systems on chips (SoCs).[3–5] These networks must provide ultralow latency and scalable, high-bandwidth on-chip communication to support a wide range of applications with diverse traffic characteristics, while adhering to tight area and power budgets with tractable hardware complexity. The state-of-the-art packet-switched on-chip network uses a modular design in which network links are shared over multiple packet flows. Although this enables high bandwidth, it comes with a significant delay, energy, and area overhead because it requires a complex router at every node. Packets must compete for resources on a hop-by-hop basis while going through a complex router pipeline at each intermediate node along their path. Thus, in such networks, contention for resources at intermediate routers dominates packet energy and delay.

As a solution to this problem, we propose *express virtual channels* (EVCs), predefined virtual express lanes in the network on which packets can bypass intermediate routers by skipping the entire router pipeline. This novel flow-control and router microarchitecture design significantly cuts down packet latency and router energy, simultaneously reducing contention and boosting throughput. The scheme pushes both performance and energy toward those of an ideal interconnection fabric, in which all communicating nodes are directly connected by pairwise dedicated wires.

Published by the IEEE Computer Society

State-of-the-art network

Figure 1a shows an example of a typical packet route in a baseline, state-of-the-art network. A packet traveling from node 01 to node 56 must go through the router pipeline at all intermediate nodes along its route. Figure 1b shows the microarchitecture of a virtual channel (VC) router present at each node in this network; its major components are the input buffers, route computation logic, VC allocator, switch allocator, and crossbar switch.

Figure 1c shows the router pipeline. The head flit of a packet, on arriving at an input port, first gets decoded and buffered according to the packet's input VC in the buffer write (BW) pipeline stage. To determine the packet's output port, route computation (RC) is performed one hop in advance using look-ahead routing,[6] thereby enabling flits to compete for VCs immediately after the BW stage. Speculation[7] further cuts down the critical path by letting the header flit simultaneously go through VC allocation (VA) and switch allocation (SA), in which it arbitrates for a VC and the switch port, respectively. If the speculation succeeds, the flit directly traverses the crossbar during the switch traversal (ST) pipeline stage. However, when speculation fails, the flit must go through these pipeline stages again, depending on where the speculation failed. This is followed by link traversal (LT), during which the flit travels to the next node. Body and tail flits follow a similar pipeline, except that they do not go through RC and VA stages, instead inheriting the VC allocated by the head flit. The tail flit, on leaving the router, deallocates the VC reserved by the header.

Under the special case of very low network load—when there are no flits ahead in the input buffers and no output port conflicts—pipeline bypassing can enable a flit to directly traverse the crossbar after a single cycle of switch setup.

Router overhead

Because a packet must go through several stages of the router pipeline before traversing the link at every hop, communication latency in packet-switched networks is dominated by the router delay, as opposed

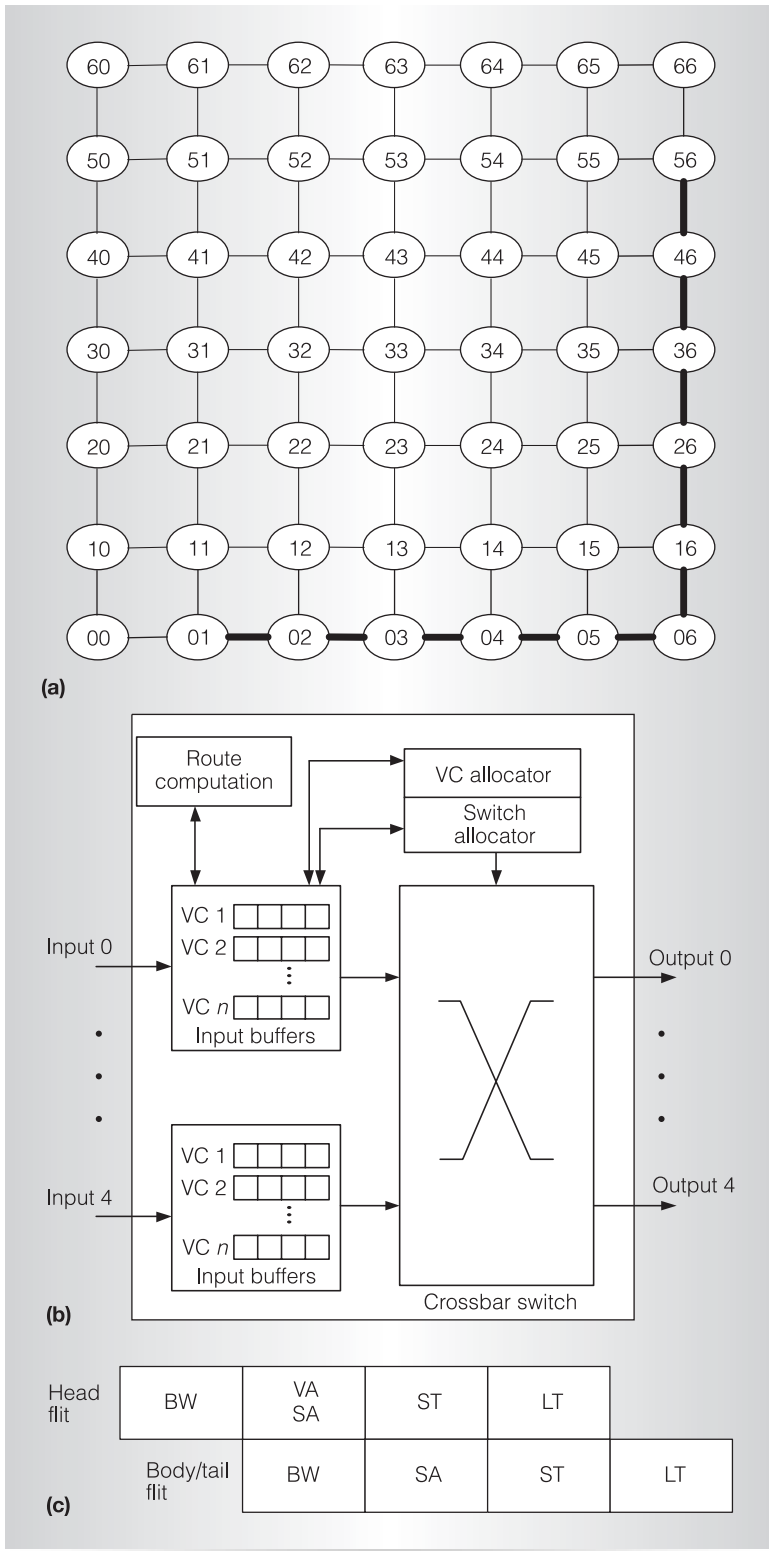

Figure 1. Baseline, state-of-the-art network: example packet route (boldface) from node 01 to 56 (a), router microarchitecture (b), and router pipeline (c).

to just the link delay. The total packet latency, T, defined as the time elapsed between the first flit of the packet being injected at the source node to the last flit being ejected at the destination, is

$$T = D/v + L/b + HT_{\text{router}} + T_c \quad (1)$$

where D is the average wire length (the Manhattan distance) between the source and destination, v the propagation velocity, L is the packet size, b is the channel bandwidth, H is the average hop-count, T_{router} is the delay through a single router, and T_c is the delay due to contention.[8]

The first term in the equation, D/v, is the time of flight or the time spent traversing the interconnect. The second term, L/b, is the serialization latency or the time for a packet of length L to cross a channel with bandwidth b. These two terms constitute the latency a packet would incur in an ideal network, in which data travels on dedicated, pipelined wires directly connecting their source and destination. However, sharing and multiplexing links between multiple source-destination flows in a packet-switched network increases latency. The third and fourth terms in equation 1—which correspond to the time spent in the intermediate routers coordinating packet multiplexing and the contention delay spent waiting for resources at intermediate hops, respectively—indicate this latency.

Multiplexing links over multiple flows in packet-switched designs also leads to additional energy consumption, with the total energy E required to transmit a packet given by

$$E = L/b(DP_{\text{wire}} + HP_{\text{router}}) \quad (2)$$

where P_{wire} is the interconnect transmission power per unit length, and P_{router} is the average router power. Again, whereas in an ideal network packets would consume only wire energy (given by the first term in equation 2), multiplexing the links over multiple flows in a packet-switched network leads to additional energy consumed at each router (the second term in equation 2). P_{router} consists of buffer read and write power, the power spent arbitrating for VCs and switch ports, and the crossbar traversal power.[9] Moreover, per-hop arbitration for router resources in a packet-switched design increases network contention (compared to an ideal network), thereby degrading overall throughput.

Express virtual channels

EVCs, which we propose in this work, are predefined virtual connections between distant nodes, which act as express lanes in the network by allowing packets to bypass the router pipeline at intermediate nodes along these lanes. Intuitively, we can achieve this design by statically designating a set of EVCs at each router that always connect nodes A and B, which are k hops away, and then prioritizing these EVCs over normal virtual channels (NVCs)[10] at the intermediate nodes. For instance, Figure 2a shows a 7 $\times$ 7 2D mesh network with 3-hop EVCs (k = 3) depicted by dotted lines. A trip from node 00 to node 03 can proceed through an EVC that virtually skips the router pipelines at nodes 01 and 02. Note that EVCs are not additional physical channels, but virtual channels (VCs)[10] that share existing physical links.

Static EVCs

We first present the details of EVCs via a static design that uses express paths of uniform lengths. Each node in a static EVC network is distinguished as either an EVC *source* or *sink node*, or a *bypass node*. A node is an EVC source or sink along a specific dimension if an EVC along that dimension originates or terminates at that node. Bypass nodes, on the other hand, do not act as EVC sources or sinks; they are the nodes that are virtually bypassed by packets traveling on EVCs. For example, in Figure 2a, node 00 is an EVC source or sink for both the x and y dimensions, whereas node 13 is an EVC source or sink node along the x dimension, and node 04 is an EVC source or sink node along the y dimension. Nodes 01 and 02 are examples of bypass nodes along the x dimension; nodes 10 and 20 are examples of bypass nodes along the y dimension.

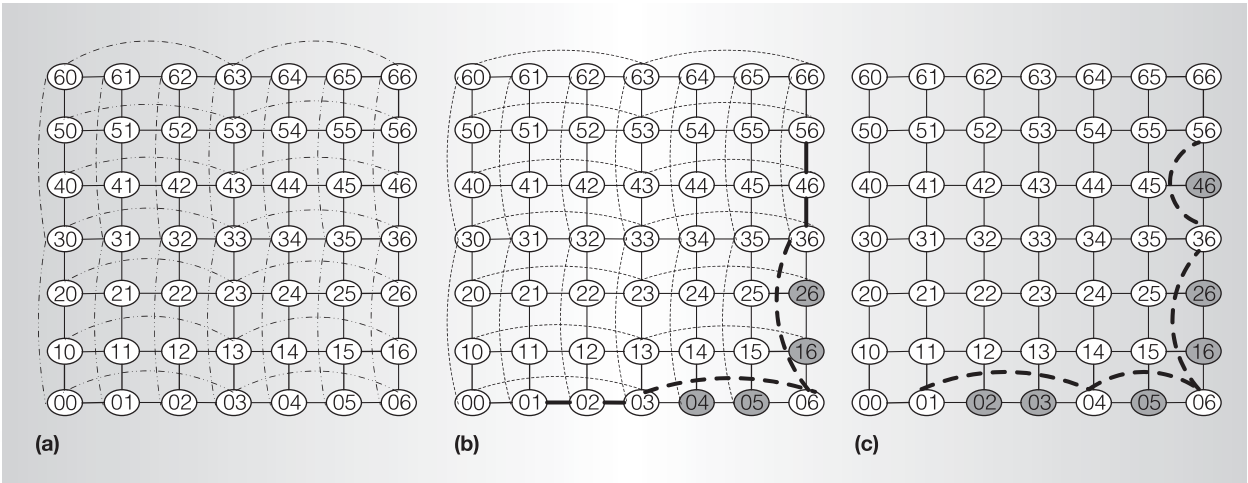

Figure 2. An EVC network (a), a 3-hop static EVC network (b), and a dynamic EVC network with maximum EVC length l_{max} = 3 (c). Solid lines are normal virtual channels; dotted lines are EVCs. In the example packet routes, shaded nodes are bypassed using EVCs.

The entire set of VCs falls into two types:

- NVCs are VCs that are allocated just as in traditional VC flow control,[10] and are responsible for carrying a packet through a single hop.
- *k*-hop EVCs are VCs that carry the packet through *k* consecutive hops, where *k* is the fixed length of the EVC and is uniform throughout the network.

Bypass nodes support the allocation only of NVCs, not EVCs; EVCs bypass their router pipelines. Therefore, packets can acquire EVCs along a particular dimension only at EVC source or sink nodes.

How are routers bypassed using EVCs? When a packet traveling on an EVC reaches a bypass node, it bypasses the entire router pipeline at that node. Figure 2b depicts the VCs acquired by a packet traveling from node 01 to node 56 using deterministic *XY* routing. Here, the packet first travels on two NVCs at bypass nodes node 01 and node 02 to reach an EVC source or sink node 03. At node 03, the packet can acquire a 3-hop EVC that originates at this node and terminates at node 06. This EVC then carries the packet through the next three hops. While traveling on this EVC, the packet skips the following operations at bypass nodes 04 and 05:

- *VC allocation.* The packet continues on the same EVC that it was allocated at node 03 and that it currently holds.
- *Switch allocation.* At each bypass node, EVC flits are prioritized over locally buffered flits. Hence, when a flit traveling in the west to east direction using the 3-hop EVC connecting nodes 03 and 06 reaches node 04, it proceeds directly to the east switch port, because it is prioritized over any local flits at node 04 also waiting to use the east output. The router at node 04 does this by nullifying the local SA result for the east output in that cycle. Hence, the EVC flit is given automatic passage without contention. A similar prioritization takes place when the EVC flit arrives at node 05.
- *Buffering.* Because EVC flits can directly use the switch port at bypass nodes without contention, they have no need of buffering at those nodes. Thus, the EVC flit starting from node 03 does not go through buffer writes or reads at nodes 04 and 05.

Because an EVC flit is prioritized over locally buffered flits at bypass nodes, it is ensured access to all output ports along the way without stalling until it reaches the EVC's endpoint node. Hence, the EVC flit

starting from node 03 and traveling west to east (Figure 2b) is ensured access to the east output upon arriving at intermediate nodes along that EVC (nodes 04 and 05) and is not stalled until it reaches the endpoint node 06, where it gets buffered and goes through the normal pipeline by contending for its next output port with other locally buffered flits. In other words, a k-hop EVC carries the packet through the next $k - 1$ nodes without stalling and having to go through the router pipeline. Thus, packets try to traverse as many EVCs as possible along their route from source to destination. Packets use NVCs only to reach an EVC source or sink and there hop onto an EVC (for example, traveling first on NVCs for two hops from node 01 to reach an EVC source or sink node 03 in Figure 2b) or when the hop count in a dimension is less than k, the EVC length. Hence, the remaining packet route involves traveling on another EVC at node 06 to reach node 36, followed by two NVC traversals to reach its destination node, 56.

How does bypassing help? Bypassing nodes using EVCs helps significantly reduce the router overhead in packet-switched designs by affecting latency, energy, and throughput.

To illustrate the latency impact, Figure 3a shows the express router pipeline through which a flit goes whenever it bypasses a node using EVCs. Because the flit does not need to go through the BW, VA, or SA stages, it can head directly to ST, followed by LT, to the next node, at the end of which the flit gets latched. The crossbar switch can be aggressively tailored for EVCs to further shorten the express pipeline by removing the ST stage and allowing EVC flits to bypass the crossbar as well (Figure 3b). This reduces the pipeline to merely link traversal, a situation approaching that of the ideal interconnect. In addition, bypassing routers using EVCs occurs nonspeculatively at all levels of network loading, unlike prior bypassing or speculation techniques, which are effective only under low network load.

Unlike speculation-based techniques, EVCs also lead to a significant reduction in network energy consumption. They do so by targeting the per-hop router energy E_{router}, which is given as

$$E_{\text{router}} = E_{\text{buffer\_write}} + E_{\text{buffer\_read}} + E_{\text{vc\_arb}} + E_{\text{sw\_arb}} + E_{\text{xb}} \quad (3)$$

where $E_{\text{buffer\_write}}$ and $E_{\text{buffer\_read}}$ are the buffer write and read energy, $E_{\text{vc\_arb}}$ is the VC arbitration energy, $E_{\text{sw\_arb}}$ is the switch arbitration energy, and E_{xb} is the energy required to traverse the crossbar switch.

While traveling on an EVC, a packet skips the router pipeline at intermediate nodes, without being buffered or having to arbitrate for a VC or the switch port. This in effect saves $E_{\text{buffer\_write}}$, $E_{\text{buffer\_read}}$, $E_{\text{vc\_arb}}$, and $E_{\text{sw\_arb}}$, thereby significantly reducing E_{router} and approaching ideal energy. The aggressive express pipeline removes E_{xb} as well, though wire energy E_{wire} increases slightly because of higher load (which we discuss later).

Given a particular topology and routing strategy, network throughput is largely determined by the flow-control mechanism. A perfect flow control efficiently uses network resources, leaving no idle cycles on the bottleneck channels. Using virtual express lanes, which effectively act as dedicated wires between node pairs, EVC-based flow control creates partial communication flows in the network, thereby improving resource use and reducing contention T_c at individual routers. Thus, packets spend less time waiting for resources at each router, which lowers the average

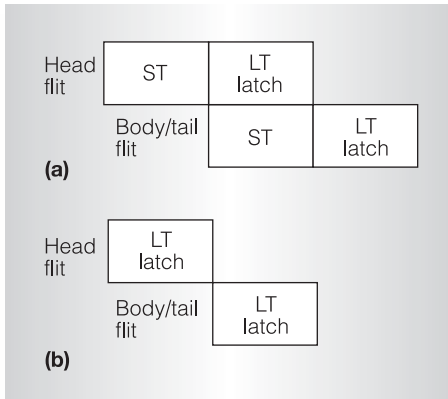

Figure 3. EVC router pipelines: express (a) and aggressive express (b).

queuing delay, allowing the network to push through more packets before saturation and hence approach ideal throughput.

Dynamic EVCs

Using static EVCs of a fixed, uniform length to connect source or sink nodes throughout the network results in a constrained, asymmetric design. First, the classification of all nodes as either bypass, or source or sink, leads to an asymmetry in the design. Packets originating at bypass nodes must first travel on NVCs before they can acquire an EVC at a source or sink node. Moreover, static EVCs lead to a nonoptimal use of express paths when packet hop counts do not match the static EVC length k. In this case, packets end up bypassing fewer nodes along their route. *Dynamic EVCs* overcome these problems by

- making every node in the network a source or sink, thereby leading to a symmetric design with fairness among nodes; and
- allowing EVCs of varying lengths (from 2 to l_{max} hops) to originate from a node, which improves adaptivity by allowing packets to pick EVCs of appropriate lengths to best match their route.

Unlike static EVCs, which partition all VCs between two bins of NVCs and uniform-length EVCs, dynamic EVCs divide the VCs at each router port into a total of l_{max} bins, with one bin for NVCs and $(l_{max} - 1)$ bins for EVCs of lengths from 2 through l_{max} hops.

Figure 2c shows the VCs acquired by a packet traveling from node 01 to node 56 using *XY* routing in a dynamic EVC network with $l_{max} = 3$, where all nodes are sources and sinks of 2- and 3-hop EVCs (for clarity, the figure shows only the used EVCs). The packet can bypass more nodes along its path by using the combination of EVCs that best matches its route.

EVC path restrictions

While connecting nodes using the virtual express lanes, EVCs are restricted to connect nodes only along straight paths in a single dimension and cannot be allowed to turn. This restriction is required to avoid conflict scenarios between flits arriving simultaneously at a node on different EVCs and asking for the same output port to turn to a different dimension. Figure 4a shows a potential conflict scenario if EVCs are allowed to turn: two different flits traveling on the two turning EVCs might arrive simultaneously at node 02 asking for the north output port. Because only one of them can be forwarded to the north port, this results in a conflict. Thus, EVC paths may not turn, and packets must go through the router pipeline and change VCs when turning to a different dimension.

Can multiple straight EVCs overlap?

Multiple EVCs that run straight along the same dimension can overlap, and no arbitration is needed among them. This is because multiple overlapping EVCs along the same dimension form distinct virtual paths but still share the same physical links.

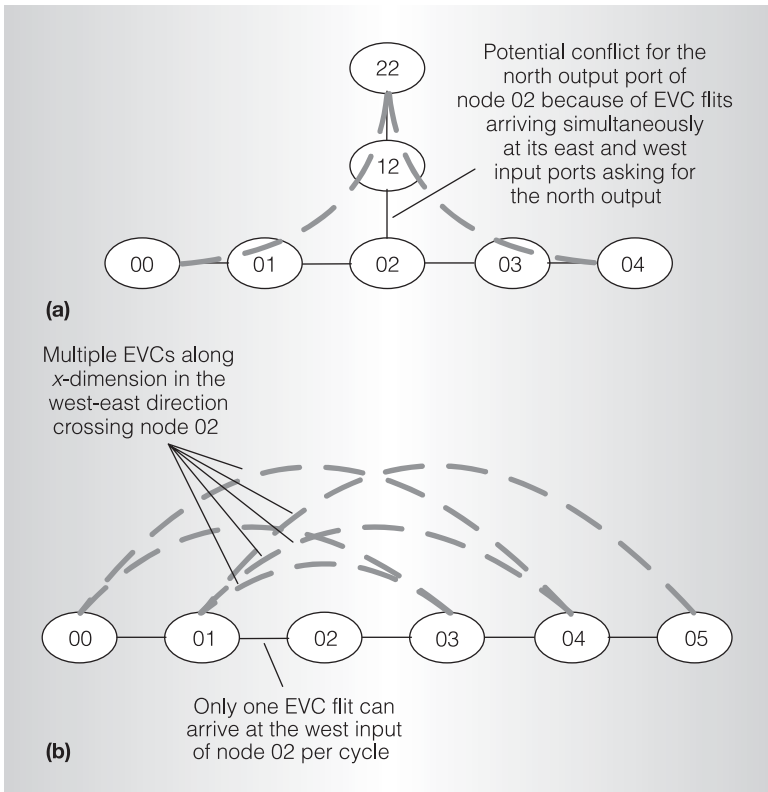

Figure 4. Multiple EVC paths: potential conflict due to turning EVCs (a) and overlapping EVCs in the *x* dimension (b).

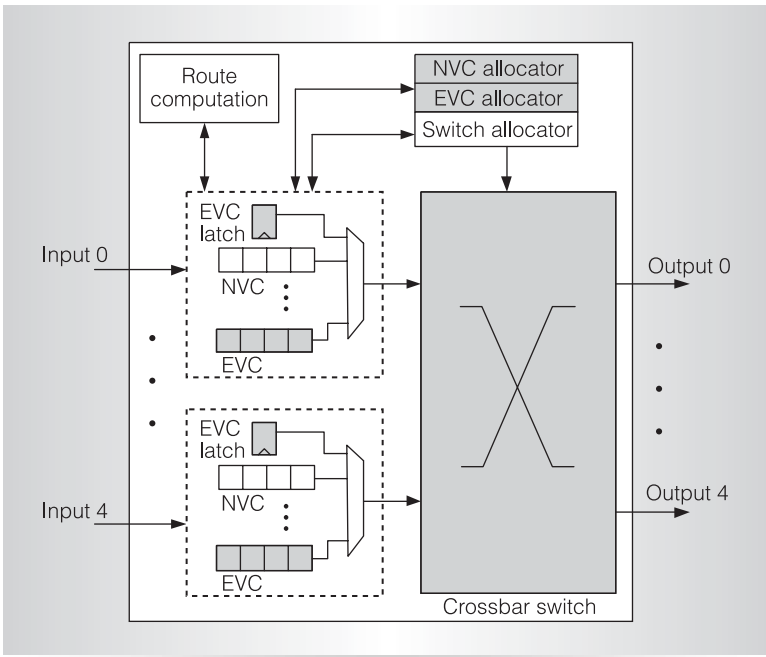

Figure 5. EVC router microarchitecture.

Hence, flits traveling on multiple overlapping EVCs along the same dimension will always arrive at any particular node sequentially, with at most one flit targeted at a particular output port every cycle. Figure 4b further illustrates this idea, showing multiple EVCs along the *x*-dimension crossing node 02 in the west-east direction, in an $l_{max} = 4$ dynamic EVC design in which each node is a source or sink of 2-, 3- and 4-hop EVCs. As shown, even though multiple flits might be using different EVCs that cross node 02 in the west-east direction, they all share the same west input link to enter node 02. Hence, from the perspective of node 02, only one EVC flit can arrive at its west input (asking for the east output) each cycle. Thus, node 02 can simply forward this flit to its east output by prioritizing it over locally buffered flits. Hence, there is no need for arbitration among overlapping EVC paths.

EVC implementation

We turn now to the implementation details of EVCs.

Router microarchitecture

Figure 5 shows the microarchitecture of a router in a dynamic EVC design. The shaded portions of the figure indicate its differences from a generic router, which we discuss in the following paragraphs.

An *EVC latch* is used at each input port to hold flits arriving on an EVC. As mentioned earlier, EVC flits do not need to be buffered at the router while bypassing.

Virtual channel allocators. As mentioned earlier, the entire set of VCs at each port in a dynamic EVC design are partitioned into NVCs and EVCs of lengths from 2 through l_{max} hops. To prevent head-of-line blocking, each node uses two separate sets of VC allocators: one allocates EVCs; and the other, NVCs. Depending on the output port and number of hops left in the packet's next dimension, the packet places a request either to allocate an NVC (if the number of straight hops left in the next dimension is less than 2, the smallest EVC length) or to allocate an appropriate EVC based on the number of straight hops left in the next dimension.

Switch design. The nonaggressive express pipeline in Figure 3a requires no modifications to the crossbar switch design, because the EVC latch in the input port is multiplexed with the local NVC input buffers, sharing a single input port to the crossbar. However, the aggressive express pipeline in Figure 3b, in which EVC flits bypass the switch altogether, requires additional links around the crossbar only along straight directions (since EVCs are only along straight paths), with these links multiplexed with the normal crossbar output. The EVC latch, in this case, must be physically located near the center of the router and acts as a staging latch, breaking the flit's path through the network and effectively bypassing the router's entire data path.

Reverse wiring. EVCs require extra wires in the reverse direction for flow-control and starvation signaling. This overhead, however, is only a small fraction of the forward wiring: As compared to a 3 percent overhead for the baseline design (assuming 128-bit-wide forward channels), the overhead for an EVC-based design is 5 percent

for static EVCs, 7 percent for dynamic EVCs with $l_{max} = 2$, and 14 percent for dynamic EVCs with $l_{max} = 3$.

Because EVC connections are virtual rather than physical, they can connect each node in the network to many other nodes using express paths, without incurring the correspondingly high area or energy overhead of additional router ports. Hence, EVCs can be implemented with low hardware overhead, using skinny and area- and energy-efficient routers and requiring only a small overhead in reverse wiring.

Buffer management

Buffered flow-control techniques require a mechanism to manage buffers and communicate their availability between routers. Since a k-hop EVC creates a virtual lane between nodes that are k hops apart, it assumes that buffer availability information is communicated across k hops so that flits are ensured of a free buffer at the downstream EVC sink node. To reduce buffer overhead and improve buffer use, we use a dynamically managed buffer design in which buffers at each router port are shared between all EVCs and NVCs. Flow control is performed using threshold-based management, by which a node sends a *stop token* to an upstream node to which it is connected (either virtually through EVCs or physically through NVCs) when the number of free buffers falls below a precalculated threshold, Thr_k for an EVC of length k. On receiving this token, the upstream node stops sending flits to the corresponding downstream node. Conversely, as downstream buffers are freed and their number exceeds Thr_k, a *start token* is sent upstream to signal restart. To ensure freedom from deadlocks, one buffer slot is reserved for each VC. This ensures that a packet can make progress even if there are no buffers left in the free buffer pool. The value of Thr_k is calculated on the basis of the hop distance between communicating nodes, which for a virtual path depends on the corresponding EVC length k, and is given by

$$Thr_k = c + 2k - 1 \qquad (4)$$

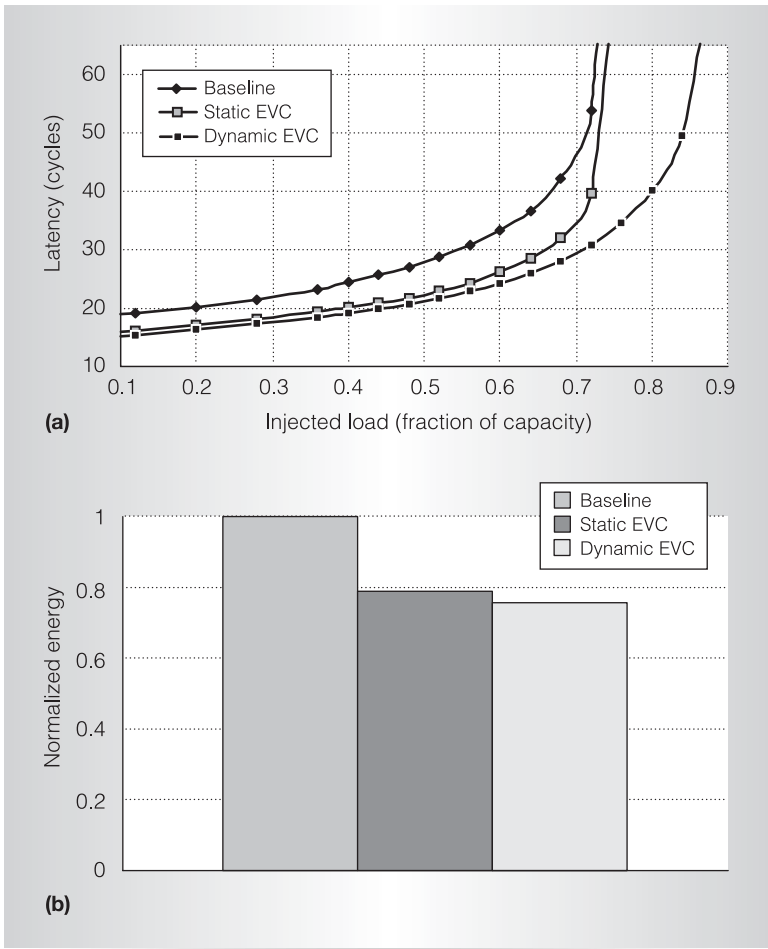

Figure 6. Uniform random traffic results: network performance (a) and router energy (b).

where c is the number of cycles taken by the token to propagate to the upstream EVC source, while there can be a maximum of $2k - 1$ flits in flight that have already left the EVC source and need to be ensured of a buffer downstream (assuming the express EVC pipeline and that token processing takes 1 cycle). In a dynamic EVC design, multiple threshold values are used corresponding to each EVC length, with longer EVC paths turned off first, followed by smaller EVCs, and finally NVCs; the reverse happens as buffers become free.

Evaluation

We evaluated EVCs using a range of synthetic traffic and SPLASH benchmark traces while exploring the design space in

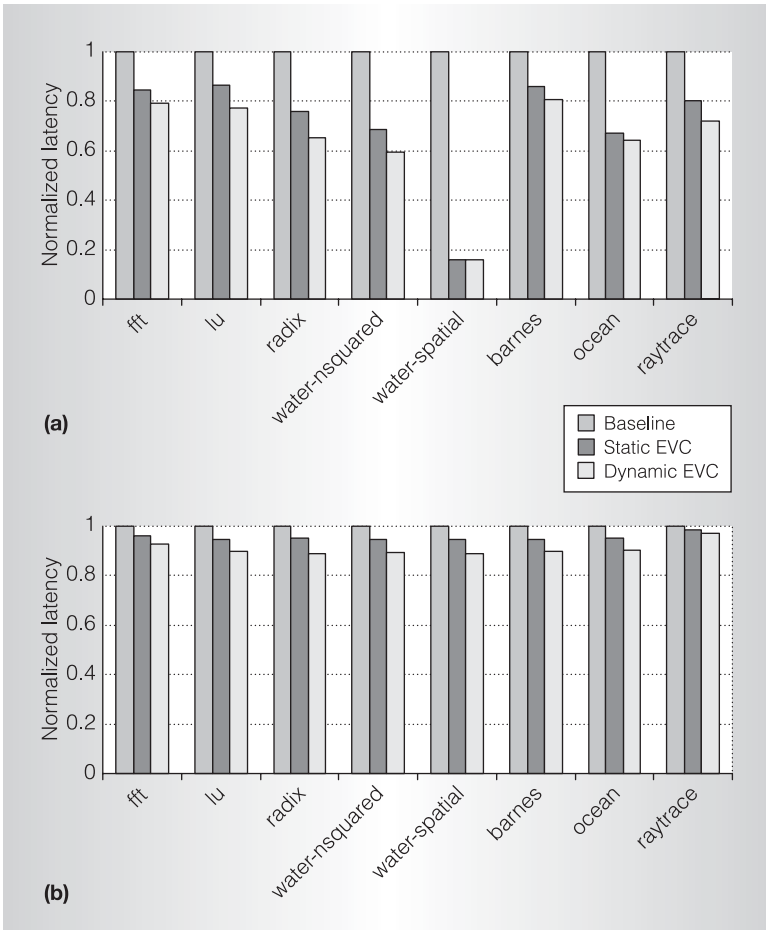

Figure 7. SPLASH benchmark results: normalized latency (a) and normalized energy (b).

depth. We evaluated network performance using an in-house commercial cycle-accurate microarchitecture simulator that models all major components of the router pipeline at clock granularity, and we used Orion,[9] an architecture-level network power model, to evaluate network energy.

Synthetic traffic

Figure 6a plots flit latencies for uniform random traffic for a 7 × 7 network as a function of the injected load (assuming 2-hop EVCs and the aggressive express pipeline). EVCs significantly outperform the baseline, with static EVCs reducing latency by 29.2 percent before the baseline saturates. The corresponding reduction for dynamic EVCs is 44.7 percent; they also offer a significant improvement in throughput, with the network saturating at around 82 percent capacity. In a comparison of router energy consumption (Figure 6b), at 70 percent capacity before the baseline saturates, static EVCs reduce router energy consumption by 21 percent, and dynamic EVCs reduce it by 24.5 percent over a power-optimized baseline. For a larger 10 × 10 network using 3-hop EVCs, the reductions are even higher: 34.4 percent and 52.8 percent in latency, and 23.5 percent and 38 percent in router energy for static and dynamic EVCs, respectively. With dynamic EVCs, the larger network also garners a 23 percent throughput improvement (approaching 88 percent capacity), which highlights the highly scalable nature of this design.

SPLASH benchmark traffic

Figure 7 plots normalized delay and router energy for a range of Splash-2 benchmarks for a 7 × 7 network, assuming 2-hop EVCs with the aggressive express pipeline. EVCs outperform the baseline across all benchmarks, reducing latency by as much as 84 percent, and router energy by as much as 11 percent.

A detailed evaluation of EVCs, using both synthetic and actual workloads, shows that EVCs significantly improve network energy, delay, and throughput as compared to a state-of-the-art packet-switched network, while requiring only minor changes to the router microarchitecture. EVCs are highly scalable and hence suitable for future CMP and SoC designs with a large number of communicating nodes. They also provide a rich and flexible design space in the form of dynamic EVCs with varying lengths, which can be used to optimize the network for specific design points. For example, the number of EVCs originating/terminating at certain nodes, as well as their lengths, can be adjusted based on the heterogeneity of the design, or based on specific traffic characteristics (if known at design time). EVCs can also be used to provide quality of service in the network with very little extra cost. By regulating the use of EVCs and associated buffers, certain traffic classes can be favored over others

(depending on service levels) by provisioning them more express paths. Similarly, latency-critical traffic can be favored to use EVCs, which, for instance, can be useful in cache-coherent CMPs where read-request messages can be prioritized in using EVCs while noncritical write-through traffic can use the normal flow. Moreover, EVCs as such provide a flow-control mechanism that is orthogonal to network topology and routing, and hence can be easily extended to different topologies such as fat trees, concentrated mesh, tori, and so on, and can be complemented by various routing optimizations. MICRO

Acknowledgments

We thank William J. Dally of Stanford University for useful feedback on this work. We also thank Ted Tabe and David V. James of Intel for their help with the modeling infrastructure and their comments on the microarchitecture. This work was supported in part by the MARCO Gigascale Systems Research Center, the Alfred P. Sloan Research Foundation, a grant from Intel, an Intel PhD fellowship, and NSF grants CNS-0613074 and CPA-0702110.

References

1. K. Sankaralingam et al., ''Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 03), IEEE CS Press, 2003, pp. 422-433.
2. M.B. Taylor et al., ''Evaluation of the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 04), IEEE CS Press, 2004, pp. 2-13.
3. L. Benini and G. De Micheli, ''Networks on Chips: A New SoC Paradigm.'' *Computer*, vol. 35, no. 1, Jan. 2002, pp. 70-78.
4. W.J. Dally and B. Towles, ''Route Packets Not Wires: On-Chip Interconnection Networks,'' *Proc. 38th Design Automation Conf.* (DAC 01), ACM Press, 2001, pp. 684-689.
5. J.A. Kahle et al., ''Introduction to the Cell Multiprocessor,'' *IBM J. Research and Development*, vol. 49, no. 4 and 5, 2005, pp. 589-604.
6. M. Galles, ''Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip,'' *Hot Interconnects 4*, 1996, pp. 141-146.
7. L.-S. Peh and W.J. Dally, ''A Delay Model and Speculative Architecture for Pipelined Routers,'' *Proc. Int'l Symp. High-Performance Computer Architecture* (HPCA 01), IEEE CS Press, 2001, pp. 255-266.
8. W.J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, 2004.
9. H.-S. Wang et al., ''Orion: A Power-Performance Simulator for Interconnection Networks,'' *Proc. Int'l Symp. Microarchitecture* (MICRO 02), IEEE CS Press, 2002, pp. 294-305.
10. W.J. Dally, ''Virtual-Channel Flow Control,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 90), ACM Press, 1990, pp. 60-68.

Amit Kumar is a PhD student in the Department of Electrical Engineering at Princeton University. His research interests include interconnection networks, computer architecture, and power- and thermal-aware system design. He has an MA in electrical engineering from Princeton University. He was awarded the Intel Foundation PhD Fellowship in 2006 and is a student member of the IEEE and ACM.

Li-Shiuan Peh is an assistant professor of electrical engineering at Princeton University. Her research focuses on low-power interconnection networks, on-chip networks, and parallel computer architectures. She has a PhD in computer science from Stanford University. She was awarded the Anita Borg Early Career Award in 2007, Sloan Research Fellowship in 2006 and the NSF CAREER award in 2003.

Partha Kundu is a researcher within Intel's Microprocessor Technology Labs in Santa Clara, California. His research interests include on-chip networks, memory system design, transactional memory, and performance simulation. He has an MS from the State University of New York, Stony Brook.

Niraj K. Jha is a professor in the Department of Electrical Engineering at Princeton University. His interests are in nanotechnology, CAD, computer architecture, security, and testing. He has a PhD in electrical engineering from the University of Illinois at Urbana-Champaign. He is a Fellow of both the IEEE and the ACM.

Direct questions and comments about this article to Amit Kumar, B217, Engineering Quadrangle, Princeton University, Princeton NJ 08544; amitk@princeton.edu.

For more information on this or any other computing topic, please visit our Digital Library at http://computer.org/csdl.