An Interconnect Architecture for Networking Systems on Chips

Network processor systems on chips meet the speed and flexibility requirements of next-generation Internet routers. The Octagon on-chip communication architecture, with its cost, performance, and scalability advantages, supports these network processor SOCs.

Faraydon Karim
Anh Nguyen
STMicroelectronics

Sujit Dey
University of California, San Diego

To meet the demands of ever-increasing Internet traffic, the next generation of Internet backbone routers must deliver ultrahigh performance over an optical infrastructure. At the current Internet traffic growth rate, network service providers will likely deploy OC-768 routers in the foreseeable future. At the same time, as Internet and application service providers attempt to provide more diverse and differentiated services, routers will have to take on new tasks. In addition to routing and packet forwarding, routers will likely perform packet classification, distinguishing packets and grouping them according to their requirements; buffer management, determining buffer allocation and admission control for packets; and packet scheduling, determining how to sequence packets to meet service level agreements (SLA).[1]

Traditionally, routers have used general-purpose reduced-instruction-set computer (RISC) processors or application-specific ICs (ASICs). Although general-purpose, processor-based router architectures provide the flexibility to upgrade to new router tasks, they will not satisfy the growing speed requirements for new, complex, packet-processing tasks. On the other hand, ASIC-based router implementations can provide the speed but not the required programming flexibility. These shortcomings of traditional RISC and ASIC designs mean that designers must develop new high-speed network processors that permit flexible programmability and work at OC-768 speed.

At OC-768 (40 Gbps), IP packet arrival rate could reach approximately 114 x 10^6 packets per second (assuming 44 bytes per packet). To ensure that the worst-case time to process a packet does not exceed the packet arrival rate and thus violate SLAs, packet-processing time should be at most 9 ns per packet. To accommodate this requirement, a network processor must perform approximately 500 instructions on each arriving packet to enable packet forwarding and classification on packet flows. Hence, an OC-768 network processor must process 57 billion instructions per second, a performance level a multiprocessor system-on-a-chip (SOC) architecture can provide.

Octagon is a novel on-chip communication architecture that can meet the performance requirements of network processor SOCs. Octagon's cost, performance, and scalability advantages make it suitable for the aggressive on-chip communication demands of not only networking SOCs, but also SOCs in several other domains.

0272-1732/02/$17.00 © 2002 IEEE

High-performance communications

Consider the on-chip communication requirements typical network processor applications impose. Using T.V. Lakshman and D. Stiliadis' packet classification algorithm with an estimated 10,000 classification rules and 16-bit on-chip memory width,[2] we must perform 625 memory accesses per packet arrival, or 71.3×10^9 memory accesses per second (in the worst case). Clearly, this necessitates the use of multiple memory components, and an on-chip communication architecture that enables highly concurrent, high-speed communication between the multiple processor and memory components.

Recent studies have demonstrated the significant role an on-chip communication architecture plays in determining a SOC's overall performance.[3] Several techniques let us design and synthesize on-chip communication to satisfy components' interface and communication needs in an application-specific system.[4–6] However, because one of a network processor's primary goals is to efficiently execute multiple applications (including evolving networking applications), synthesizing an application-specific interconnect architecture for a network processor SOC will not work.

Rather than synthesize a custom on-chip interconnect architecture for a given application, K. Lahiri and colleagues propose to optimally map a system's communication requirements to a given communication architecture.[7] In other work, they describe a technique that allows reconfiguration of the selected communication architecture's protocols according to the application's changing communication demands.[8] Proposed communication mapping and reconfiguration techniques provide up to an order of magnitude improvement in system performance. Taken together, mapping and reconfiguration techniques show promise for efficiently mapping multiple applications to the same interconnect fabric.[7,8] For these techniques to be successful, however, developers must select the appropriate on-chip communication architecture.

Despite recent advances in the analysis and design of high-performance on-chip communication architectures, commercial SOCs commonly use simple bus-based topologies and protocols.[9,10] Even the Virtual Socket Interface Alliance's efforts have focused on bus interface standards.

Although bus-based on-chip communication might be suitable for many applications, it clearly cannot satisfy an OC-768 network processor's very demanding on-chip communication needs. For high-performance computing systems, interconnect architectures based on crossbars, or crossbars mixed with buses, deliver the ultrahigh-performance communication needed among components.[11]

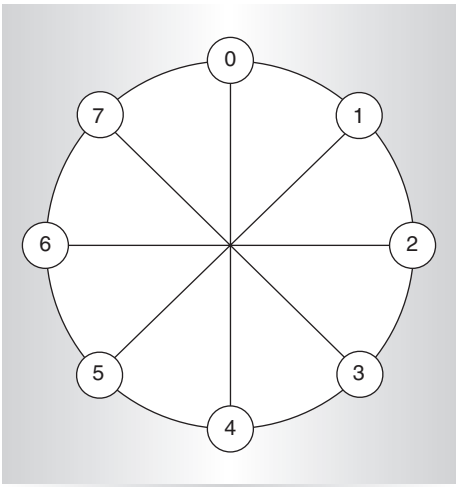


Figure 1. Basic Octagon configuration includes eight nodes and 12 bidirectional links.

Many switching architectures in high-performance routers also use crossbars.[1] An on-chip crossbar can satisfy the on-chip communication needs of an OC-768 network processor SOC. Theoretically, crossbar performance (in terms of throughput or delay) is high enough to permit development of efficient network processing tasks.[12] In reality, crossbar implementation costs are high: Crossbars require many on-chip wires and relays to minimize clock skew across the chip. In addition, crossbars do not scale well as the number of nodes to be connected increases. Although crossbar-based interconnects might be justified in high-performance computing systems and routers, they might not be the best economic choice for lower-cost and higher-volume network processor SOCs.

Octagon

The Octagon on-chip architecture is simpler to implement than a crossbar yet has much higher throughput than either shared buses or traditional crossbars. Unlike crossbars, Octagon's implementation complexity increases linearly with the number of nodes—processor or memory components—that the network must connect.

Architecture

As Figure 1 shows, a basic Octagon unit consists of eight nodes and 12 bidirectional links.

The Octagon architecture has several desirable properties:

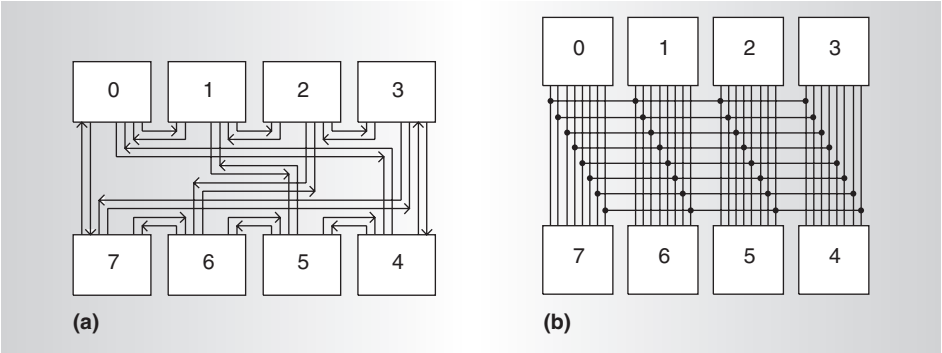


Figure 2. Octagon (a) and crossbar (b) physical-layout schematic examples. Octagon consists of 12 horizontal and 12 vertical 32-bit tracks with each horizontal track upper-bounded by 8 mm, and each vertical track upper-bounded by 0.156 micron (13 x 12 micron). The crossbar has eight horizontal and 32 vertical 32-bit tracks, with the horizontal tracks upper-bounded by 8 mm as in the Octagon, and the vertical tracks upper-bounded by 0.108 micron (9 x 12

- two-hop communication between any pair of nodes;
- higher aggregate throughput than a shared bus or crossbar interconnect under certain implementation conditions;
- simple, shortest-path routing algorithm; and
- less wiring than a crossbar interconnect.

Octagon operates in packet- or circuit-switched mode. An Octagon packet is data that must be transferred from the destination Octagon node to the source Octagon node as a result of a communication request by the source node. An Octagon packet can be fixed or variable length. In packet-switched mode, the network nodes buffer packets at intermediate nodes if there is contention at the egress link.

In circuit-switched mode, a network arbiter allocates the entire path between source and destination nodes of a communicating node pair for a number of clock cycles. Nonoverlapping communication paths can occur concurrently—that is, the arbiter permits spatial reuse. In this mode, system performance is a function of the chosen connection schedule. The question is, then, given the set of pending communication requests, how should the arbiter schedule connections to optimize throughput (or some other metric)? We have developed a simple connection scheduler, called the best-fit algorithm (described later), to enable Octagon's circuit-switched mode.

Packet routing

We can code Octagon node addresses into a three-bit field and route an Octagon packet as follows. We prepend a three-bit tag to each packet. Each node compares the tag (Packet_addr) to its own address (Node_addr) to determine the next action. The node computes the relative address of a packet as

```
Rel_addr = Packet_addr – Node addr (modulo 8)
```

At each node on the Octagon, packet routing is a function of Rel_addr:

- *Rel_addr* = 0, process at node
- *Rel_addr* = 1 or 2, route clockwise
- *Rel_addr* = 6 or 7, route counterclockwise
- Route across otherwise

Consequently, a predetermined, simple routing scheme for each network packet permits at most two hops to separate any two nodes.

Implementation cost

Figure 2 illustrates the physical layout of the Octagon and crossbar interconnect architectures. In our network processor, each Octagon node consists of a processor-memory pair with an estimated size of 2 mm × 2 mm. Let us assume that the minimum wire spacing is 0.2 μm, and the width of a 32-bit link is 12μm (including individual wire width, spacing, and shielding). As Figure 2a shows, the

Octagon architecture consists of 12 horizontal and 12 vertical 32-bit tracks. Each horizontal track is upper-bounded by 8 mm, the total width of the four nodes. Each vertical track is upper-bounded by 0.156 micron (13 × 12 micron). As Figure 2b shows, the crossbar needs eight horizontal and 32 vertical 32-bit tracks. Although the horizontal tracks are upper-bounded by 8 mm as in the Octagon, the vertical tracks are upper-bounded by 0.108 micron (9 × 12 micron). Thus, wiring in Octagon is less complex than in a crossbar.

Octagon versus crossbars and buses

Consider a typical SOC communication. Node processes continuously generate requests for service; examples include memory read and write requests. We classify requests according to their source-destination pair; thus, we denote requests that originate from node i with destination node j as type ij. Requests of type ij arrive at the system following the Poisson process with parameter λ_{ij}, which is the requests' arrival rate. Service time is the time a destination node requires to complete all requested tasks if it processes the request in isolation. We assume the communication links between source and destination nodes are locked until service is completed.

Service and response times

The required service time is equivalent to packet size or link rate, where packet size is the number of bytes of data transfer that result from the communication request, and link rate is the communication link's data transfer rate. We assume that the service time for request ij is exponentially distributed with parameter μ_{ij}, with $1/\mu_{ij}$ as the average service time per request—a reasonable assumption because packet length varies and, in the most general case, could range from one to thousands of bytes. For example, if a node issues a read request for an 8-byte data block, then for a 1-MHz 8-bit wide data bus, the request service time is 8 microseconds.

We can easily extend these assumptions to accommodate other discrete-timed distribution such as Bernoulli arrivals, and geometric or deterministic service time. We consider a symmetric system where $\lambda_{ij} = \lambda$, $\forall i,j$ and $\mu_{ij} = \mu$, $\forall i\, j$. The utilization of requests type ij is $\lambda_{ij} = \lambda_{ij}/\mu_{ij} = \lambda/\mu = \lambda$. Utilization is the average amount of service demand arriving within one time unit. The aggregated arrival rate is $\lambda_{tot} = S_{ij}\,\lambda_{ij}$. Total utilization $\rho_{total} = \lambda_{tot}/\mu$.

We model the shared bus as a single server queue with Poisson arrivals and exponential service time. Consider the aggregated request arrival process for the eight nodes. Because the individual arrival process is a Poisson distribution, the superposition is also a Poisson distribution.[12] In memory access applications, service rate corresponds to memory access speed. We ignore all propagation delays. The server serves queued requests in first-in, first-out (FIFO) order. An arriving request's response time is the difference between the arrival time and time the bus completes the service. Because the server is work conserving,[13] the expected response time, denoted by EW_{bus}, for an arbitrary request arriving at the single server queue is identical to that for a single-server multiple-queue system. The expected response time of a shared bus modeled by a single-server queue is

$$EW_{bus} = \frac{\rho_{tot}}{\lambda_{tot}\left(1-\rho_{tot}\right)}.$$ [13]

For crossbar throughput, we use the model presented by J. Chen and T. Stern.[12] They showed that for a large switch (approximately 20 nodes) having a speedup factor of one, response time is

$$EW_{xbar} = EW_s + \frac{\lambda_{tot}E^2W_s}{2\left(8-\lambda_{tot}EW_s\right)},$$

$$\text{where } EW_s = \frac{8\lambda_{tot}}{\left(8\mu-\lambda_{tot}\right)^2}p_o + \frac{1}{\mu},$$

$$\text{and } p_o = 1 - \frac{\rho_{tot}}{8}.$$

Chen and Stern also investigated the impact of various arbitration policies on switch performance. They found that arbitration policies do not affect maximum throughput because it is only a function of the average service. Different arbitration policies result in different response times, however.

Communication request scheduling

We investigate the Octagon architecture performance through simulation, using a simple request-response traffic model. That is, a source

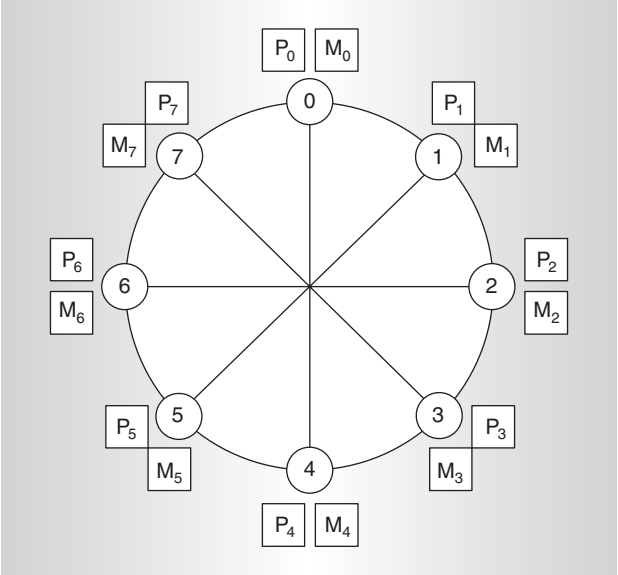


Figure 3. High-level application model. Each node is associated with a processor and a memory module.

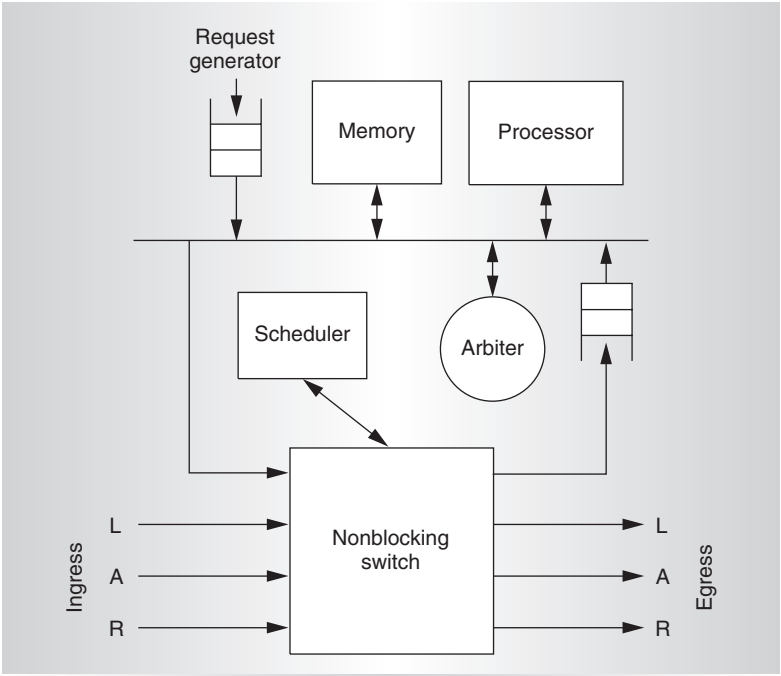


Figure 4. Node model. Each node has a request generator, processor, memory, and three ingress and three egress ports. Switch arbitration is through the central scheduler.

node generates a request to send to a destination node. It eventually establishes a connection for the communication. For each connection, the source node sends a request and receives a response. After the communications, the participating nodes sever the connection.

We associate a processor and memory module with each node, as Figure 3 shows. Applications of this traffic model exist in routing table lookup, Internet protocol packet classification, and other networking functions where each node generates memory access requests. If the requested memory location is attached to the local node, then it generates no Octagon communication requests. Otherwise, it must forward the request to the appropriate node via Octagon using the routing algorithm previously presented. At the destination node, the memory request consumes several clock cycles before spawning a response, which it returns to the originating node.

The best-fit scheduler is a connection-oriented communication protocol that can simultaneously accommodate nonoverlapping connections. Each node maintains three queues of outstanding requests, one for each egress link. With respect to the overall network, this *global* scheduler gives priority to the head-of-line requests in arrival time order (lower arrival time implies higher priority). At every service completion time, the scheduler checks to see if it can make new connections based on the previously described priority scheme. The scheduler sets up connections until it can accommodate no more without violating the nonoverlapping rule. Note that we only consider head-of-line requests at each node. When a connection is torn down, the scheduler reactivates to check if it can set up new connections.

Figure 4 is a detailed view of the node model. In addition to the request generator, processor, and memory, each node has three ingress and three egress ports labeled left, across, or right, consistent with its associated neighbor. Logically, the node emulates a simple 4×4 nonblocking switch (plus processor and memory). The switch has neither input nor output buffering. The central scheduler performs switch arbitration.

Performance results

Figure 5 compares the expected response time of Octagon, a bus, and a crossbar. We fix $\mu = 0.5$ per clock cycle and vary λ_{tot}. The horizontal axis represents $\rho_{tot} = \lambda_{tot}/\mu$, and the vertical axis represents expected response time in clock cycles. Recall that λ_{tot} is the system's total packet arrival rate; $1/\mu$, its average packet size;

and ρ_{tot}, average number of packets the system can service concurrently. Octagon has significantly higher maximum throughput than both the bus and crossbar. We obtain similar results for fixed-size packets.

As these results show, the bus saturates at $\rho_{tot} = 1$ because a single server (the bus bandwidth) can provide at most one service unit per time unit. For the crossbar, we assume a single queue per crossbar node—hence we can model the crossbar as a system of eight queues sharing eight servers. Contention and head-of-line blocking mean that the eight available servers provide approximately four work units per time unit.[12] Therefore, crossbar saturation occurs at $\rho_{tot} \cong 4$. We could have considered eight queues per crossbar node, but the implementation cost would be prohibitive.

On the other hand, it might be reasonable for each Octagon node to have three queues. Hence we model the Octagon architecture as a system of 24 queues and 24 servers (three egress queues and three outgoing links per node). This means we incur more cost per node for Octagon than for the crossbar. However, saturation for Octagon occurs at $\rho_{tot} \cong 12$, which is significantly higher than for a crossbar, as Figure 5 shows. Note that the effective server utilization is about 50 percent (12), the same as for the crossbar.

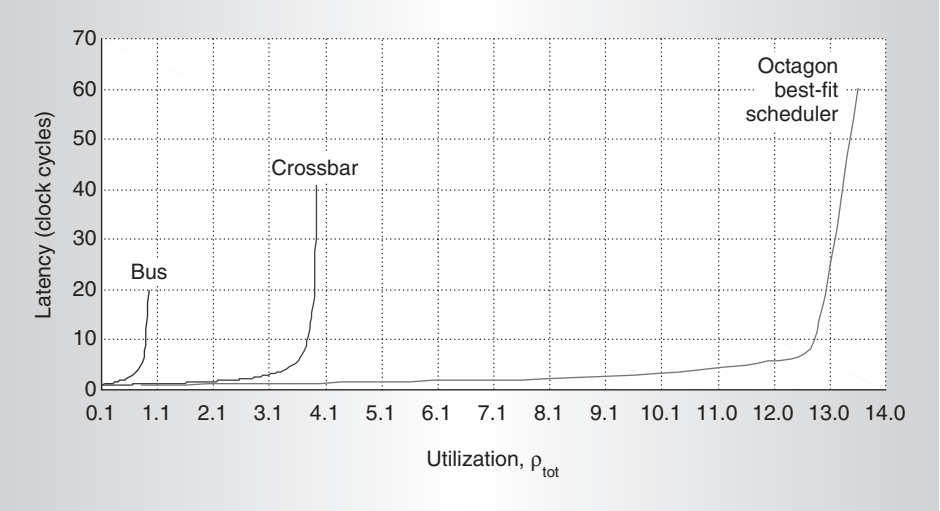


Figure 5. Throughput comparison of Octagon, a bus, and a crossbar for randomized packet

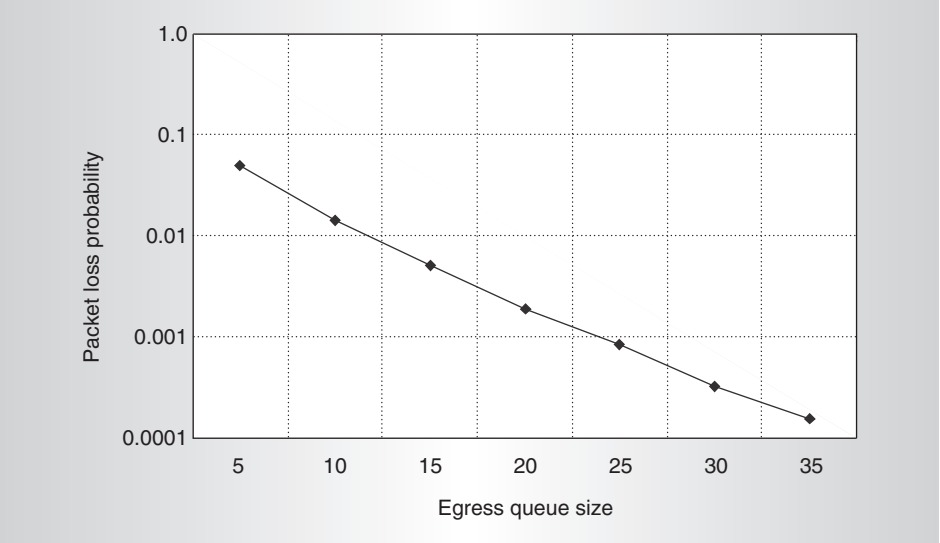
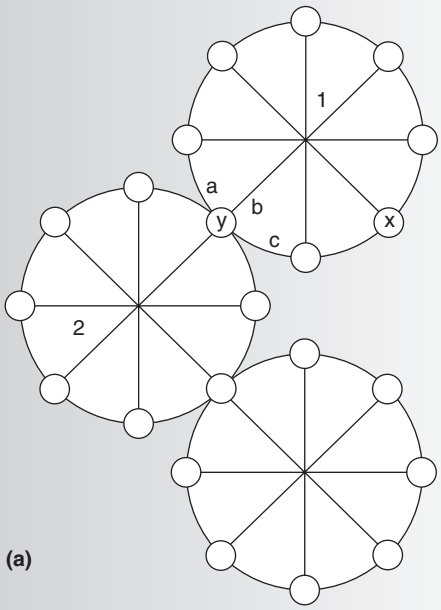


Figure 6. Packet loss probability versus egress queue size for a system using the Octagon architecture.

Some packet service approaches achieve high throughput by compromising service latency. That is, system efficiency and throughput increase as the workload at each queue builds. Figure 6 shows that at relatively high utilization of $\rho_{tot} = 12$ and at 10^{-4} packet loss probability, a system using an Octagon architecture requires fewer than 50 packet buffers. Thus, the average queue occupancy is not excessive.

For Octagon's best-fit connection scheduling, a node (process) is not blocked if the scheduler cannot schedule its communication request immediately. Instead, the requesting node queues the request in its egress queue. This strategy can improve system performance and node utilization more than some communication protocols (especially most bus protocols), which stall the requesting node until its request can be granted. However, each Octagon node must have a queue large enough to avoid packet loss. Figure 6 shows the packet



Octagon

Nodes	Horizontal links no./ max length (mm)	Vertical links no./ max length (mm)	Maximum distance (hops)
8	12/8	12/0.156	2
15	24/8	24/0.156	4
22 (shown)	36/8	36/0.156	6 (shown)

(b)

Crossbar

Nodes	Horizontal links no. /max length (mm)	Vertical links no. /max length (mm)	Maximum distance (hops)
8	8/8	32/0.108	1
15	15/16	120/0.192	1
22	22/11	242/0.276	1

(c)

Figure 7. Scaling strategy 1. Bridge nodes (node *y*) connect adjacent Octagons (a) and perform hierarchical packet routing. Member nodes attach to only one Octagon (node *x*). The tables give the maximum distance for Octagon (b) and crossbar (c) networks of various sizes.

loss probability as the size of each node's egress queue increases. Note that a queue size of 25 results in a nominal packet loss of 0.1 percent, while a queue size of 35 reduces the packet loss probability to 0.01 percent. If needed, a system designer can enable a zero packet loss guarantee in Octagon by having the packet scheduler refuse requests if the egress queue is at full or near-full capacity, thereby stalling the requesting node (as many existing buses do).

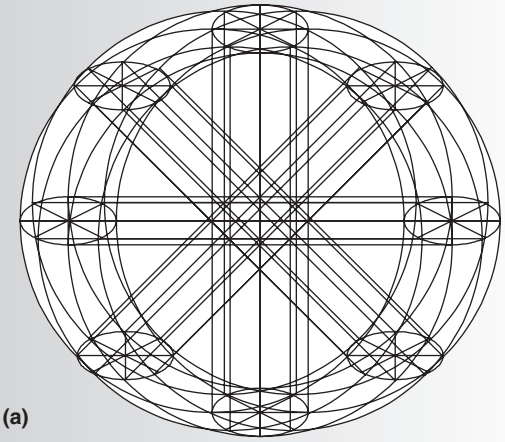
Scalability

The increasing performance demands of programmable network processors makes scalability an important factor in Octagon's design. Next-generation network processors will likely have 16 or more processors and many distributed memory components holding tables for Internet protocol lookup and classification. The need for interconnecting greater numbers of on-chip components in network processors and other SOCs will accelerate in the foreseeable future, increasing the need for scalable, on-chip communication architectures.

Strategy 1: Low wiring complexity

One of the Octagon architecture's strengths is its ability to scale linearly. Figure 7a shows a scaling strategy that requires two different node types: bridge and member. As the name implies, bridge nodes connect adjacent Octagons and perform hierarchical packet routing (for example, node *y* in Figure 7a). Member nodes attach to only one Octagon (node *x*, for example). Consider a network consisting of eight interconnected Octagons. The Octagon address field of each packet is 6 bits wide: three high-order bits to identify the local Octagon and three low-order bits to identify the node within the Octagon. Each bridge node performs static routing based on the entire field, and each member node performs routing based only on the three low-order bits.

Figure 7 also shows the estimated wiring cost as a function of increasing network size for the Octagon and crossbar architectures. To arrive at these estimates, we assumed the SOC layout in Figure 2a for each octagon. We extended the layout to the configuration of Figure 2b for crossbars with more than eight nodes. Octagon scales linearly and the crossbar does not because in the crossbar, every node is wired to every other node. As the number of nodes *N* grows, the number of required wires is of



(a)

Nodes	Links	Maximum distance (hops)
8	12	2
64 (shown)	192	4 (shown)
512	2,304	6

(b)

Figure 8. Scaling strategy 2. We extend the Octagon to the multidimensional space by linking corresponding nodes in adjacent Octagons according to the Octagon configuration (a). The table (b) indicates the increased wiring complexity (number of links), and the greater maximum distance needed as the number of connected nodes increases.

order $O(N^2)$. In this Octagon scaling strategy, on the other hand, each node requires either three or six wires to its neighbors, resulting in wiring complexity of $O(cN)$.

The tables in Figure 7 shows the maximum distance (the maximum number of hops between any two nodes) for networks of various sizes. The maximum distance increases linearly as Octagon grows, while the crossbar has a constant maximum distance irrespective of the number of nodes. Although a higher maximum distance can degrade performance, the performance results in Figure 5 indicate that this might not always be the case: An 8-node Octagon with maximum distance 2 performs better than an 8-node crossbar with maximum distance 1.

In this strategy, the maximum distance between nodes grows much more slowly, but it does not remain constant as for the crossbar. This is fine for SOCs where low wire complexity is the dominant consideration. However, this characteristic might not suit systems where high throughput is the primary concern. For example, consider a SOC with 15 nodes. A bridge node connects Octagons 1 and 2. If all network traffic is across networks, then a bottleneck occurs at the bridge because it must transmit all traffic from Octagon 1 over the three links a, b, and c. Therefore, it can concurrently transfer at most three packets in each direction, one measure of a communication architecture's maximum throughput.

Strategy 2: High performance

For systems in which high performance is the dominant consideration, we propose a second scaling strategy that performs better than the first but has more complex wiring. In this strategy, we extend Octagon to multidimensional space. Figure 8a illustrates this scaling strategy in a 64-node Octagon. We index each SOC node by the 2-tuple (i, j), $i, j \in [0, 7]$. For each $i = I$, $I \in [0, 7]$, we construct an Octagon using nodes $\{(I, j), j \in [0, 7]\}$, which results in eight individual Octagon structures. We then connect these Octagons to each other by linking corresponding i nodes according to the Octagon configuration. That is, each node (I, J) belongs to two Octagons: one consisting of nodes $\{(I, j)\ j \in [0, 7]\}$, and the other consisting of nodes $\{(i, J)\ i \in [0, 7]\}$. The table in Figure 8 indicates the increase in wiring complexity (number of links) and maximum distance (number of hops) as the number of connected nodes increases. The maximum distance between nodes scales much better under strategy 2 than strategy 1. However, strategy 2's better performance scalability comes at the cost of greater wiring complexity.

Figure 9 (next page) shows a natural scaling approach for networks with fewer nodes, and therefore fewer Octagons to connect (each node represents an Octagon). We scale the network as follows. To connect two Octagons, we construct a link between corresponding nodes; this bidirectional link connects node i from

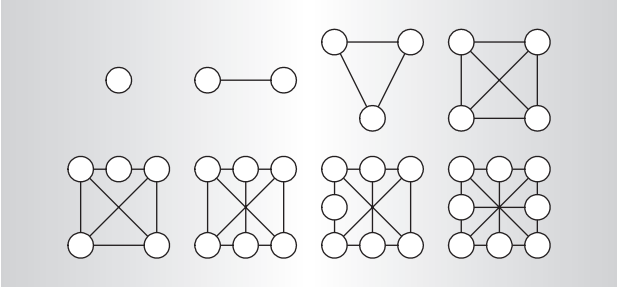


Figure 9. Growing the network using scaling strategy 2.

Octagon 1 to node *i* from Octagon 2. As the number of Octagons to be connected increases, we link corresponding nodes according to the Octagon rule. That is, to connect eight Octagons, 12 bidirectional links are needed to connect each node *i* of Octagons 0, 1, 2, …, 7. For a network with more nodes, we start adding links in a new dimension. As Figure 9 shows, by increasing the network size we can maintain the maximum hop count at three while increasing the number of nodes to 32. These nodes represent a network of four Octagons with two hops to the corresponding intra-Octagon node and one hop to the destination Octagon.

Figure 10 shows Octagon's advantage over the crossbar architecture as the number of nodes increases. Although the crossbar's implementation cost (measured in number of links) increases prohibitively with increasing nodes, the Octagon scaling strategies make scaling feasible.

Our analysis shows that Octagon significantly outperforms shared bus and crossbar on-chip communication architectures in terms of performance, implementation cost, and scalability. We are currently investigating the use of Octagon to satisfy the on-chip communication needs of other application-specific multiprocessor SOCs. MICRO

Acknowledgments

We acknowledge Naresh Soni for his encouragement and support, and Razak Hossain for his help on physical layout and implementation issues.

References

1. V.P. Kumar, T.V. Lakshman, and D. Stiliadis, "Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet," *IEEE Comm.*, vol. 36, no. 5, May 1998, pp. 152-164.
2. T.V. Lakshman and D. Stiliadis, "High-Speed Policy-Based Packet Forwarding Using Efficient Multidimensional Range Matching," *Proc. ACM SIGCOMM*, ACM Press, New York, 1998, pp. 191-202.
3. K. Lahiri, A. Raghunathan, and S. Dey, "Evaluation of the Traffic Performance Characteristics of System-on-Chip Communication Architectures," *Proc. 14th Int'l Conf. VLSI Design*, IEEE CS Press, Los Alamitos, Calif., 2001, pp. 29-35.
4. J.A. Rowson and A. Sangiovanni-Vincentelli, "Interface-Based Design," *Proc. 34th Ann.*

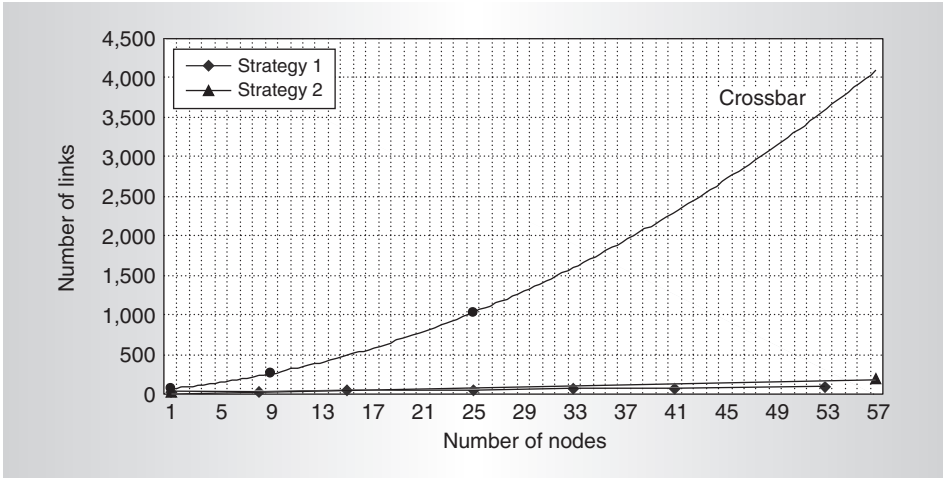


Figure 10. Comparison of Octagon scaling strategies to a crossbar architecture: number of nodes versus wiring complexity.

Design Automation Conf., ACM Press, New York, 1997, pp. 178-183.
5. R.B. Ortega and G. Borriello, "Communication Synthesis for Distributed Embedded Systems," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 98), IEEE CS Press, Los Alamitos, Calif., 1998, pp. 437-444.
6. M. Gasteier and M. Glesner, "Bus-Based Communication Synthesis on System Level," *ACM Trans. Design Automation Electronic Systems*, vol. 4, no. 1, Jan. 1999, pp. 1-11.
7. K. Lahiri, A. Raghunathan, and S. Dey, "Communication Architecture Tuners: A Methodology for the Design of High-Performance Communication Architectures for System-on-Chips," *Proc. 37th Design Automation Conf.*, ACM Press, New York, 2000, pp. 513-518.
8. K. Lahiri, A. Raghunathan, and S. Dey, "Efficient Exploration of the SOC Communication Architecture Design Space," *Proc. Int'l Conf. Computer-Aided Design* (ICCAD 00), IEEE CS Press, Los Alamitos, Calif., 2000, pp. 424-430.
9. Sonics Integration Architecture, www.sonicsinc.com.
10. D. Flynn, "AMBA: Enabling Reusable On-Chip Designs," *IEEE Micro*, vol. 7, no. 4, July-Aug. 1997, pp. 20-27.
11. A. Charlesworth, "The Sun Fireplane Interconnect," *IEEE Micro*, vol. 22, no. 1, Jan.-Feb. 2002, pp. 36-45.
12. J. Chen and T. Stern, "Throughput Analysis, Optimal Buffer Allocation, and Traffic Imbalance Study of a Generic Nonblocking Packet Switch," *IEEE J. Selected Areas in Comm.*, vol. 9, no. 3, Apr. 1991, pp. 439-449.
13. D. Gross and C. Harris, *Fundamentals of Queueing Theory*, 3rd ed., John Wiley & Sons, New York, 1998, pp. 297-300.

Faraydon Karim is an ST Fellow at STMicroelectronics' Advanced System Technology, Advanced Computing Lab in La Jolla, California. His research interests include computer and embedded system architecture. Karim has a PhD in computer engineering from La Salle University. He is a member of the IEEE.

Anh Nguyen is a research engineer at the STMicroelectronics Advanced Systems Technology, Advanced Computing Lab in La Jolla, California. His research interests include performance analysis, resource allocation, and algorithm development. Nguyen has a PhD in electrical engineering from the University of California, Los Angeles. He is a member of the IEEE.

Sujit Dey is a professor in the Electrical and Computer Engineering Department at the University of California, San Diego. His research interests include configurable platforms consisting of adaptive wireless protocols and algorithms, and deep-submicron adaptive SOCs for next-generation wireless appliances and network infrastructure devices. Dey has a PhD in computer science from Duke University. He is a member of the IEEE.

Direct questions and comments to Faraydon Karim at STMicroelectronics, Advanced System Technology, San Diego, CA 92121; faraydon.karim@st.com.