

---

# A FOUR-TERABIT PACKET SWITCH SUPPORTING LONG ROUND-TRIP TIMES

---

THIS 4-TBPS PACKET SWITCH USES A COMBINED INPUT- AND CROSSPOINT-QUEUED (CICQ) STRUCTURE WITH VIRTUAL OUTPUT QUEUING AT THE INGRESS TO ACHIEVE THE SCALABILITY OF INPUT-BUFFERED SWITCHES, THE PERFORMANCE OF OUTPUT-BUFFERED SWITCHES, AND LOW LATENCY.

**François Abel**  
**Cyriel Minkenberg**  
**Ronald P. Luijten**  
**Mitchell Gusat**  
**Ilias Iliadis**  
IBM Research, Zurich  
Research Laboratory

..... The primary goal of packet switches used in communications networks is to optimize the aggregate throughput performance; in computer interconnect networks, the emphasis is on minimizing latency while providing features such as reliable delivery. Despite other specific differences between the two types of networks, basic quality-of-service (QoS) requirements such as guaranteed delay and bandwidth delivery are more or less the same. The significant overlap in packet switch requirements in these two application domains prompts the question of whether a single architecture can satisfy both. From an economic perspective, this would clearly be desirable. Hence, we present the design of a 4-Tbps packet switch useful in single- and multistage configurations for both domains.

To meet communications networks' stringent requirements for throughput and QoS, we use a *combined input- and crosspoint-queued* (CICQ) switch architecture, a subclass of the well-known combined input- and output-queued (CIOQ) architecture. Our CICQ architecture consists of a buffered crossbar switch with a virtual output-queuing arrangement at the ingress—a combination that exhibits nearly ideal performance

characteristics, even under bursty traffic conditions.<sup>1</sup> The architecture's excellent bisectional bandwidth<sup>2</sup> and QoS characteristics also meet computer interconnect networks' strict latency requirements.

As an example of how to deploy our switch architecture, we'll describe a single-stage communications router that supports 64 to 256 ports with corresponding rates of 64 to 16 Gbps. In systems such as this one, the large number of line cards and the large switch fabric practically require that we must distribute the system over multiple racks; this results in the packing of line cards tens to hundreds of feet away from the switching core.<sup>3</sup> At a 64-Gbps port rate, the resulting round-trip time has considerable impact on the system architecture's performance. So far, this issue has received little attention. We've designed our architecture to directly support a wide range of round-trip time values in the switch fabric itself—a novel approach to the challenges of multiterabit-per-second switching fabrics. We build our system from four different CMOS ASIC building blocks, using a total of 40 chips for the switching core and 64 fabric interface chips on the line cards to achieve an aggregate throughput of 4 Tbps.

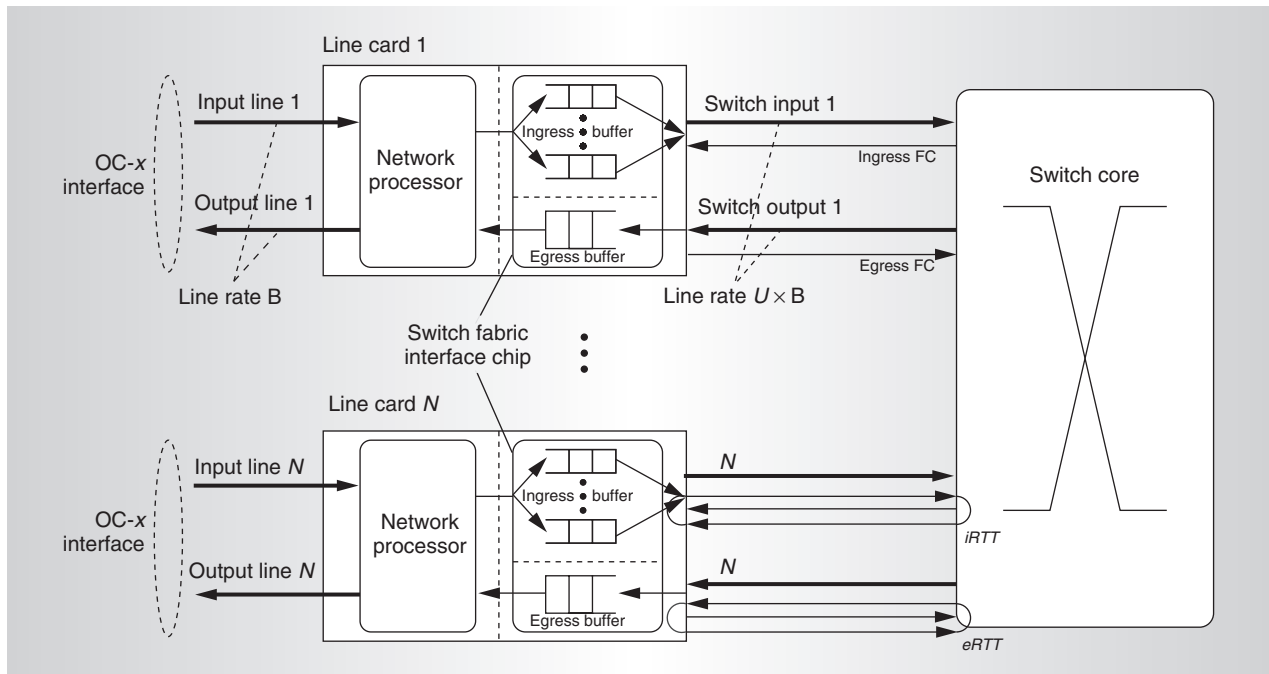


Figure 1. Single-stage packet switch: generic system architecture. A speedup of  $U$  is available between the line cards and the switch core.

### Forerunners and related work

Most present-day single-stage switch architectures use virtual output queuing—VOQ—at the ingress, as in Figure 1. The key feature differentiating such architectures is whether scheduling of the ingress VOQs is *centralized* or *distributed*.

The centralized approach typically uses  $N$  input buffers organized by destination (VOQ) combined with an  $N \times N$  bufferless crossbar switch core. Within this centralized scheduling category, we can further distinguish among approaches without speedup (which are purely input-queued) and those with limited speedup (which are CIOQ). Because the core is bufferless, this speedup applies to the ingress buffer read access, the egress buffer write access, the arbitration process, and the entire switch core, including links, as shown in Figure 1. Both approaches require a centralized arbitration algorithm to resolve input and output contention. The purely input-queued approach requires a bipartite-graph matching algorithm, such as PIM or iSLIP.<sup>4</sup> For the CIOQ approach, researchers have proposed more complex arbitration algorithms<sup>5,6</sup> to achieve exact output-queuing emulation, which produces better QoS sup-

port and performance—particularly under nonuniform traffic.

The main drawback of the centralized scheduling approach is the limited scalability of its arbitration algorithm. First, it requires a high degree of connectivity and tight coupling (synchronization) between the arbitration unit and all ingress line cards to exchange requests and grants. Second, the arbitration algorithm must converge within one packet cycle, which becomes extremely challenging at high line rates and port counts because of the arbitration algorithms' superlinear complexity.

The distributed approach removes the need for centralized input arbitration by using a limited number of output buffers with full speedup, typically integrating them in the switch core. In other words, this method creates a CIOQ architecture based on a buffered switch core with internal speedup of  $N$  and eliminates the need to speed up the links. This approach requires link-level flow control between the input and output buffers to ensure losslessness and prevent output buffer monopolization (hogging); typically, such architectures use backpressure-, grant-, or credit-based flow control.

The main drawback of distributed CIOQ is the complexity of implementing output

queuing with full speedup. The traditional implementation uses a shared-memory switch core, where all inputs and outputs simultaneously access a common memory.<sup>7-9</sup> However, with current technology this implementation does not scale to multiterabits per second. Moreover, for architectures that use VOQ at the ingress to reduce head-of-line blocking, the performance advantage of a shared memory is no longer compelling.<sup>10</sup>

This situation calls for a more distributed output buffer implementation, such as can be achieved by partitioning the shared memory into dedicated buffers for groups of inputs or outputs. Taking this approach to the extreme leads to the well-known classic buffered crossbar architecture, which provides a dedicated buffer for every input-output combination. Several groups have recently proposed to combine such a buffered crossbar with VOQ ingress buffers (CICQ) and have demonstrated performance very close to ideal output queuing.<sup>1,11-13</sup> The performance depends on the contention resolution mechanisms for the VOQs and outputs—for example, RR\_RR,<sup>1</sup> OCF\_OCF,<sup>12</sup> and LQF\_RR.<sup>13</sup> (These *XX\_YY* notations indicate the combination of scheduler disciplines, with *XX* denoting the VOQ arbitration and *YY* denoting the output queue arbitration. RR is round-robin, OCF is oldest-cell first, and LQF is longest-queue first.) However, because of practical implementation issues, the existing proposals have mostly considered only one or a few packets per crosspoint, and they have paid little attention to the interaction between round-trip time, work conservation, and QoS scheduling.

### Why CICQ?

Although the CICQ concept isn't new,<sup>14,15</sup> advances in CMOS integration density and the addition of VOQ at the ingress now make it practical. Consequently, researchers have renewed interest in this architecture;<sup>1,11-13</sup> until now, however, no one has considered CICQ in light of long round-trip time and feasibility issues.

We have several key reasons for proposing the CICQ architecture as the basis for a generic switch design for both high-speed communications and computer interconnect networks:

- Most practical, centralized scheduling algorithms have a heuristic, suboptimal nature and tend to suffer from performance deterioration under nonuniform traffic conditions.<sup>1,13</sup> CICQ switches exhibit close to ideal performance characteristics even for nonuniform traffic.<sup>1</sup> This is crucial in supporting any traffic pattern specific to the two application domains.
- For any nonblocking CIOQ architecture, whether shared-memory or buffered-crossbar, the total buffer requirement to support any traffic pattern scales quadratically with  $N$  and linearly with round-trip time:  $O(N^2 \times RTT)$ .<sup>16</sup> The increasing line speed and the increasing physical size of switch systems imply a long intrafabric round-trip time, which, combined with the growing port count, drives the buffer requirements up. Whereas a shared-memory implementation of such a large buffer does not scale beyond a 1-Tbps throughput, a buffered-crossbar implementation is practical, as we show in this article.
- In a CICQ system, when there is no contention, packets proceed immediately to the output without waiting for the end of a request-grant-accept cycle, as the case would be with a centralized arbiter. This significantly reduces latency during low load.
- The CICQ architecture distributes contention resolution among all inputs and outputs: Independent input schedulers resolve input contention, and the output buffers resolve output contention. This results in a simpler, more distributed implementation— $2N$  schedulers of  $O(N)$  complexity instead of one  $O(N^2)$  scheduler. This leverages the newer trends in CMOS technology: greater density (more parallelism) rather than increased clock speeds (faster logic).
- Decoupling the arrival and departure processes within the switch core relaxes the coordination among line cards and the core. This is particularly important for scalability because it simplifies the synchronization constraints of large switches distributed across multiple racks.<sup>3</sup>



service by means of priorities or virtual lanes (for example, to support IEEE 802.1D/Q) and implements what is known as a multilayered scheduling structure to provide differentiated delay and bandwidth guarantees at the level of individual flows. Our multilayered QoS scheduler (MLQS) consists of a network of schedulers distributed over all the contention points of the system: ingress fabric interfaces, switching core, and egress fabric interfaces. There is one QoS scheduler per contention point and each scheduler implements a work-conserving algorithm by combining strict priority and weighted round-robin scheduling. In this scheme, traffic flows with low and strict delay-jitter requirements receive strict priority scheduling and map into a high-priority service class. Traffic flows that require minimum guaranteed bandwidth but no specific delay bound receive weighted-scheduling and map into an average-priority service class. Finally, the unused bandwidth from the guaranteed flows is redistributed to best-effort flows, which receive strict priority scheduling and map into a low-priority service class. Contention resolution mechanisms enforce fairness—round-robin for the input VOQs and weighted round-robin for the crosspoint arbitration.

For lossless operation and reduced end-to-end latency, as required by computer interconnect networks such as InfiniBand, PCI Express, and Rapid IO, we employ a robust link-level credit-based flow control. This flow control protocol and a flexible queuing scheme guarantee our system to be strictly nonblocking—at the port, subport, and class-of-service levels.

To address output ports, the system uses a direct bitmap, which inherently supports multicast. It uses an additional 2-bit subport destination (SPD) identifier to address subports. To reduce overhead, we define four 16-port groups—O[00:15], O[16:31], O[32:47], and O[48:63]—encoded by a 2-bit bitmap extension (BME). This requires a 16-bit bitmap and a 2-bit extension, which easily fits in a 4-byte packet header. Performing a multicast requires up to four packets.

The fabric interface implementation is a combined ingress-egress chip that connects to any network processor unit compliant with the NPF Streaming Interface standard (NPFIS). The system segments the data

stream entering the switch fabric into fixed-sized packets of either 64 or 80 bytes. Although the system achieves 100 percent throughput without requiring speedup, it operates with a line rate escalation of 1.6 (that is, 64 Gbps instead of 40 Gbps) to compensate for the switch packet header (4 to 5 bytes) and segmentation overhead. This results in a switch fabric with aggregate throughput of 4 Tbps (2.5 Tbps at the OC-768 and OC-192 levels).

A chipset of four ASIC devices implements the switch architecture—three for the switch core and one for the fabric interface. You can use this chipset to scale from 1 to 4 Tbps by using 27 to 104 devices.

### Ingress fabric interface

Our architecture organizes VOQs per destination and per class of service. The ingress path of the fabric interface (iFI) supports 256 unicast destinations and four multicast group destinations (one per BME). Including the classes of service, this leads to a total of 2,080 VOQs— $(256 + 4) \times 8$ . The iFI implements two switch paths that connect to an active X and a redundant Y switch plane for full redundancy (1:1). It stores up to 4K incoming packets and performs lossless switch-over between active and redundant planes. The VOQ scheduler performs the QoS scheduling first and then uses the round-robin algorithm to select packets of the same class of service when multiple queues are eligible for transmission.

### Switch core

The switching core has a crosspoint queuing architecture with additional QoS support. It consists of a single-stage switch architecture with  $64 \times 64$  logical links operated at 64 Gbps. At 64 Gbps, a 64-byte packet lasts 8 ns. The subport configuration consists of the four subport streams multiplexed onto a single logical link.

The  $64 \times 256$  arrangement of the internal memory and queuing structure covers all possible port configurations, including that where each of the 64 destination ports consist of four subports.

*Logical crosspoint architecture.* We define  $XP_{i,o}$  (Figure 2) to be the logical crosspoint dedicated to physical input  $i$  and physical output  $o$ . The switch core has a total of 4,096 cross-

points ( $64 \times 64$ ), each with a separate data and control path section. Organization of the data and control sections depends on the output port configuration—that is, whether or not it supports subports. For example, in Figure 2, crosspoints corresponding to output 0 are configured in subport mode; those corresponding to output 63 are not.

We define  $XPM_{i,o}$  in Figure 2 as a logical crosspoint's data section. It implements a crosspoint memory buffer dedicated to physical input  $i$  and physical output  $o$ ; it consists of one unicast- and one multicast-dedicated area. If physical output  $o$  operates with subports, the unicast area has four further partitions, each dedicated to one subport (as in  $XP_{0,0}$  in Figure 2); this is necessary to avoid buffer monopolization at the subport level. The crosspoint control section provides the storage address for a unicast packet arriving at  $XPM_{i,o}$ . Each dedicated unicast area represents an individual flow control domain that the control section manages by means of specific credits. There are a maximum of 16,384 ( $64 \times 64 \times 4$ ) unicast flow-control domains.

We define  $XPC_{i,o}$  as a logical crosspoint's control section. This consists of an output-queuing structure that routes and queues packet addresses, an address manager that controls the pool of free address locations within the unicast areas of the data section, and a credit manager that flow controls the unicast-dedicated areas. This structure is a set of 40 output queues (corresponding to four subport queues plus one multicast queue for each of eight classes of service). It guarantees nonblocking behavior at the subport level, whereas class-of-service queuing is required for QoS scheduling at the current output contention point. If physical output  $o$  does not have subports (as in  $XP_{0,63}$  in Figure 2), then the crosspoint's control section uses only 16 of 40 queues. The switching core implements a total of 163,840 queues ( $40 \times 4,096$ ). When it is used in a computer interconnect, one can assign these queues to virtual lanes, as in InfiniBand, or as a combination of priorities and transactions, as in Rapid IO.

*XPM dimensioning.* Considering that switches and routers typically inhabit 19- to 21-inch shelves, each with approximately 16 line cards, our architecture needs at least five shelves—

one for the switch core and four for the fabric interfaces—interconnected with cables or optical fibers. This physical distance combined with the short packet duration (8 ns) causes a significant intrafabric round-trip time. This is an important new design factor that current CICQ proposals have not yet addressed extensively. We define the input round-trip time ( $iRTT$ ) as the composite of the transmission packet delay from an ingress fabric interface to a crosspoint destination plus the transmission of the flow control information back to the iFI (see Figure 1). We express  $iRTT$  as the number of packets in flight, and it includes the contribution of the cables, the logic at the receiver and transmitter sides (such as data deskewing and alignment), and the VOQ and crosspoint arbitration times. We define  $eRTT$  similarly for the egress side. Credit flow control inherently satisfies the losslessness property, but to satisfy the input work-conservation property under any traffic pattern scenario, we must scale the size of the  $XPM_{i,o}$  proportionally to  $iRTT$ , which translates into at least  $iRTT$  credits being available to fully utilize the link (memory size and credits are linear functions of round-trip time). This ensures that any ingress port can transmit to any egress port at any instant and at full rate—for example, under directed traffic or in the absence of output contention, when traffic should always proceed at the maximum rate.

Our requirement is to support any traffic pattern over a distance of 100 feet of interconnect between the fabric interface and the switch core, which translates into  $iRTT = eRTT = 64$  packets. This number of credits (64) suffices for operation over interconnect lengths much greater than 100 feet if the range of traffic patterns is limited. For example, considering only uniformly distributed traffic lets us operate with link lengths up to 10,000 feet and still achieve 100 percent throughput. Finally, to fully decouple unicast and multicast performance, we allocate  $2 \times iRTT$  packets per XPM, where the size of the multicast area is programmable between 0 and  $iRTT$  packets.

*Unicast and multicast operation.* Figure 2 shows the distributed output buffers and output queues of the buffered crossbar switch core. In this design, arriving packets from a given input port of the switch are broadcast to all cross-

points attached to that specific input port.

For unicast packets, every crosspoint in a row filters out the bitmap and the header's BME fields to determine if a packet is destined for the output to which that crosspoint connects. If so, the XPC generates a unicast store address for the XPM, which writes the packet into the crosspoint buffer memory. At the same time, the address is also queued in the XPC, according to the subport destination and class-of-service fields.

This broadcast-and-select behavior also provides multicast capability, because multiple bits set into the bitmap select multiple crosspoints. In contrast to the unicast scenario, a multicast address manager (MCAM) common to 16 output ports (one multicast group) generates the same multicast store address for all the XPMs of the multicast group. This guarantees that all the replicated copies of a multicast packet are written to the same address within all the dedicated multicast areas; it also implies that a multicast group constitutes one flow control domain. The use of a unique multicast address simplifies the design of the MCAM, which generates and recycles these addresses. The XPC has one specific multicast queue per class of service.

*Crosspoint arbitration.* Each output port has one crosspoint arbiter. Every 8 ns, the arbiter selects one of the  $64 \times 40$  queues using the following algorithm:

1. Mask out the destinations that have either no traffic or no credit available for the corresponding subport or multicast destination.
2. Select a subport or multicast destination according to the round-robin pattern: SP0, MC, SP1, MC, SP2, MC, SP3, MC.
3. For the selected subport or multicast destination, select a class of service according to the QoS scheduler of that output port.
4. Among the crosspoints that have traffic for the selected destination (step 2) and for the selected class of service (step 3), select one according to a weighted round-robin schedule.

Steps 2 and 4 are spatial within the domains of subports, unicast and multicast, and crosspoints; step 3 is temporal.

### Egress fabric interface

Owing to the reduced memory requirement, driven only by the link-level flow control ( $eRTT = 64$ ), we can implement the egress fabric interface (eFI) in the same chip as the ingress (see Figure 1 and Figure 2). If the configuration enables subports, the egress buffer has five partitions—one per subport and one for multicast. Otherwise, the egress buffer has two areas—one for unicast and one for multicast. We replicated this scheme for the two planes (X and Y). Queuing occurs per switch plane, subport destination, and class of service, resulting in 64 queues ( $2 \times 4 \times 8$ ). At port level, the switch core handles multicast; at subport level, the eFI expands multicast through a routing index that addresses a multicast table.

### VLSI implementation

Now we'll demonstrate the feasibility of our CICQ system by describing its implementation into a chipset of four different ASICs. Because of space limitations, we focus mainly on the switch core, the system's most challenging part.

### Cost and power requirements

To keep the system cost-effective, we limit the die size to a maximum of  $250 \text{ mm}^2$  and the pin count to approximately 1,000 signal I/Os (for a total of about 1,500 pins per package). To keep design time short, we consider only standard-cell design methodologies, processes, and packages.

To comply with Network Equipment Building Standards (NEBS) rules, a single board must not dissipate more than 150 W. At the same time, single-chip power dissipation must remain under 25 W to avoid hot spots and remain forced-air cooled.

### Sizing

We based our sizing and design assumptions solely on currently available, proven technologies. Our target CMOS technology is 0.11-micron copper (Cu-11). In calculating the sizing, we assumed that all the design's critical parts are VHDL-coded and synthesized. These critical parts include a specific 4-input  $\times$  4-output cluster that implements the control section of 16 logical crosspoints, an equivalent  $4 \times 4$  cluster for the data section implementation, and the crosspoint arbiter.

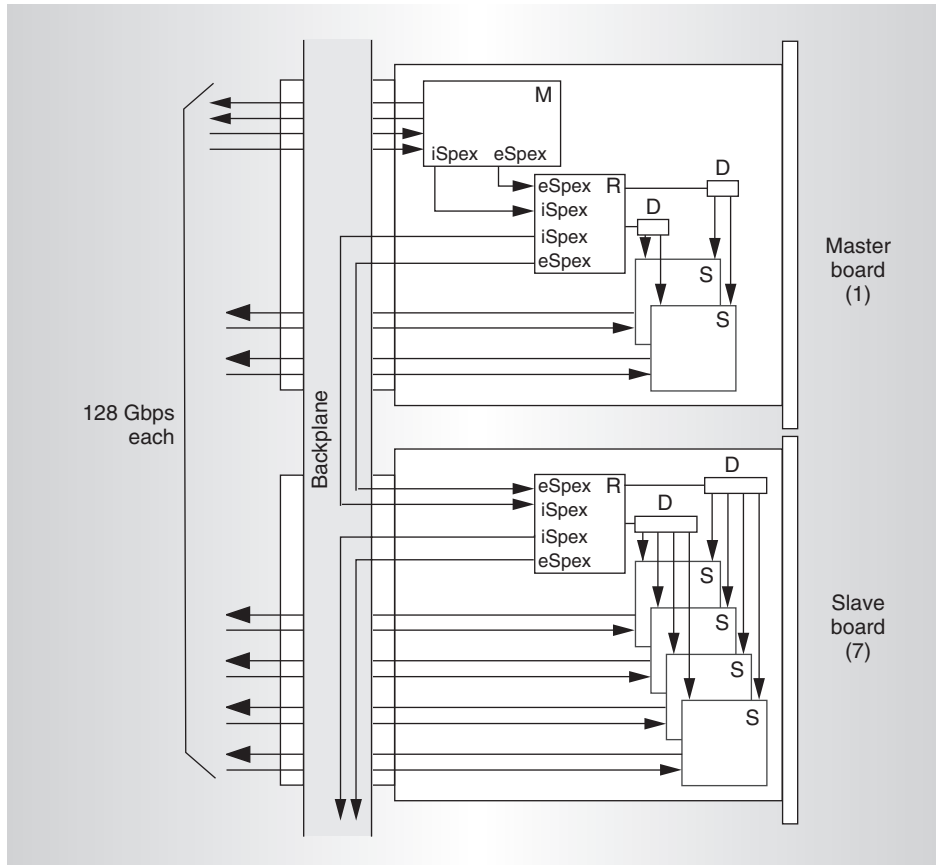


Figure 3. Physical switch core implementation. D—driver; M—master chip; R—redriver chip; S—slave chip.

For the digital and analog hard macros—such as memories and serializers/deserializers—we retrieved sizing figures from the standard-cell ASIC library. We consulted previous switch designs to estimate all the other chiplets individually. Finally, we accounted for a wiring overhead of 100 percent for all control logic except for the hard macros, which are already placed and routed.

### Physical implementation

In addition to the quadratic memory challenge, I/O pins and I/O power requirements constrain the design of a buffered crossbar. (The I/O limitation is inherent in any crossbar—buffered or bufferless.) Given the high-speed and high-density serializers/deserializers available in ASIC libraries today (roughly 125 mW per 2.5 to 3.2 Gbps full-duplex in devices from Fujitsu, IBM, LSI, Lucent, and Texas Instruments), a 4-Tbps switch requires about 10,000 pins and dissipates roughly 256 W of

I/O power, thus requiring at least 16 chips to meet the pin and power constraints.

To reduce the serialization latency as required by computer interconnect networks and to overcome the I/O and memory limitations of buffered crossbars, we expand the port rate and buffering capacity by stacking multiple (31) switch slices operated in parallel and using a single master chip to control them. This technique of parallel, sliced switching has proved successful in scaling shared-memory architectures up to 1 Tbps;<sup>9,10</sup> it is often called port speed expansion (Spex) mode or distributed switch architecture with centralized control. We based the architecture described here on a single switch domain; it therefore does not provide multiple paths. However, our 4-Tbps switch core can serve as a building block with external load-balancing controllers to realize an  $N \times N$  multipath switch with a higher external port rate.

Figure 3 shows our proposed implementa-

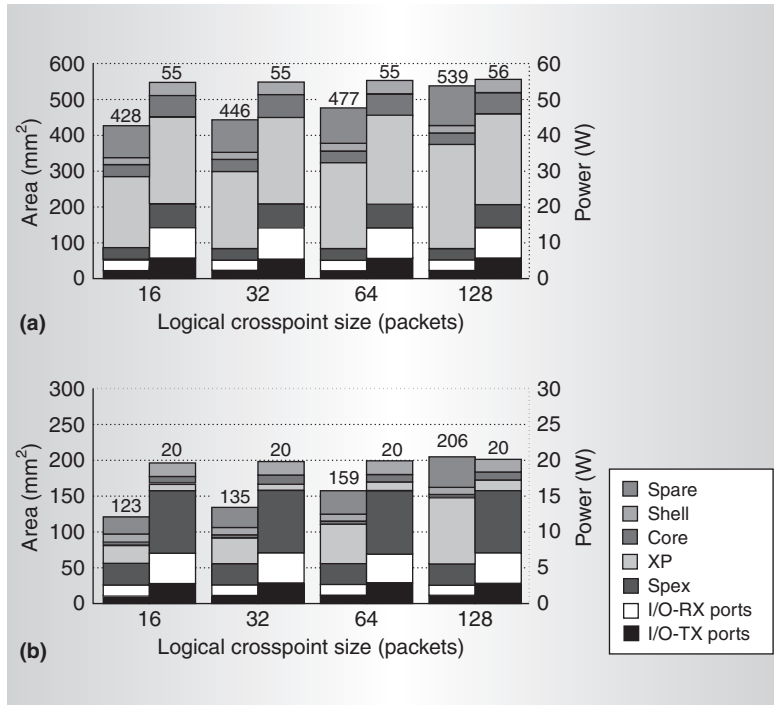


Figure 4. Master (a) and slave (b) chip sizing.

tion of the switching core based on the Spex concept distributed over eight physical boards. It consists of one master chip (labeled M in Figure 3) and 30 slave chips (labeled S) organized in a combination chain- and tree-based topology. The master implements the control sections (XPCs) of the 4,096 crosspoints, and each of the 30 slaves implements 1/30 of the data sections (XPMs).

The iFI partitions incoming packets into  $k$  = 32 segments of 16 bits (or 20 bits, if the packet size is 80 bytes) before sending them to the switching core over  $k$  different links, each operating at 2.5 Gbps. The first two segments, containing the packet header, go to the master module; the  $k-2$  other segments, containing only data payload, go to the slave modules. When the master receives its two segments, it extracts the header information and queues the store address according to the routing and QoS information that the header carries. At the same time, 2 bytes of control information—comprising the destination XPM within a row, the store address within that XPM, and a unicast/multicast flag—goes to the slaves over the ingress speed expansion interface (iSpex). This information tells the slaves how to handle their data segments.

(Another reason that multicast packets share the same store address is to reduce the iSpex interface's bandwidth.)

Every time the master schedules a packet for transmission, it must retrieve the corresponding data payload segments from all slave chips. Therefore, a similar process takes place at the egress side of the switch core over an egress speed expansion interface (eSpex). The control information transmitted on this interface specifies an XPM selection within a column and a retrieve address within the selected XPM.

Each card contains one redriver chip (labeled R in Figure 3) to reshape the iSpex and eSpex interface signals (also implemented with high-speed serializers/deserializers) before forwarding them via a backplane to the next daisy-chained card. Driving the signals without reshaping takes place at board level using a commercially available driver (labeled D).

The package for the 4-Tbps switch core is eight physical cards, each consuming a single slot of a 19- to 21-inch shelf. Every card connector handles a bisectonal bandwidth of 1 Tbps (512 Gbps in and 512 Gbps out), which translates into a minimum card edge of 18 inches, given the state-of-the-art 2.5-Gbps connectors and plug-compatible cable assemblies. Connecting the eight 1-Tbps cards over a backplane requires approximately 4,000 differential wire pairs and is only feasible by keeping the wires short (one inch) and spatially localized. We achieve this by plugging in the eight switch core blades only every other slot (for example, slots 0, 2, 4, ..., 14), thus always leaving an entire slot free for the interconnect between line cards and switch core.

### Master and slave chip sizing

Figure 4 shows the area use and power consumption of the master and slave chips. The  $x$ -axes of the two graphs indicate four logical crosspoint sizes, corresponding to packet round-trip times of  $RTT/4$ ,  $RTT/2$ ,  $RTT$ , and  $2 \times RTT$ . The seven parts of each stacked bar represent the area contributions and power consumption of the serializers' (TX) and deserializers' (RX) I/O interface, the serializers' or deserializers' Spex interface, the logical crosspoints (XPM or XPC), additional control for the switch core (core) and the switch I/O protocol (shell), and spare area for the global wiring (spare).

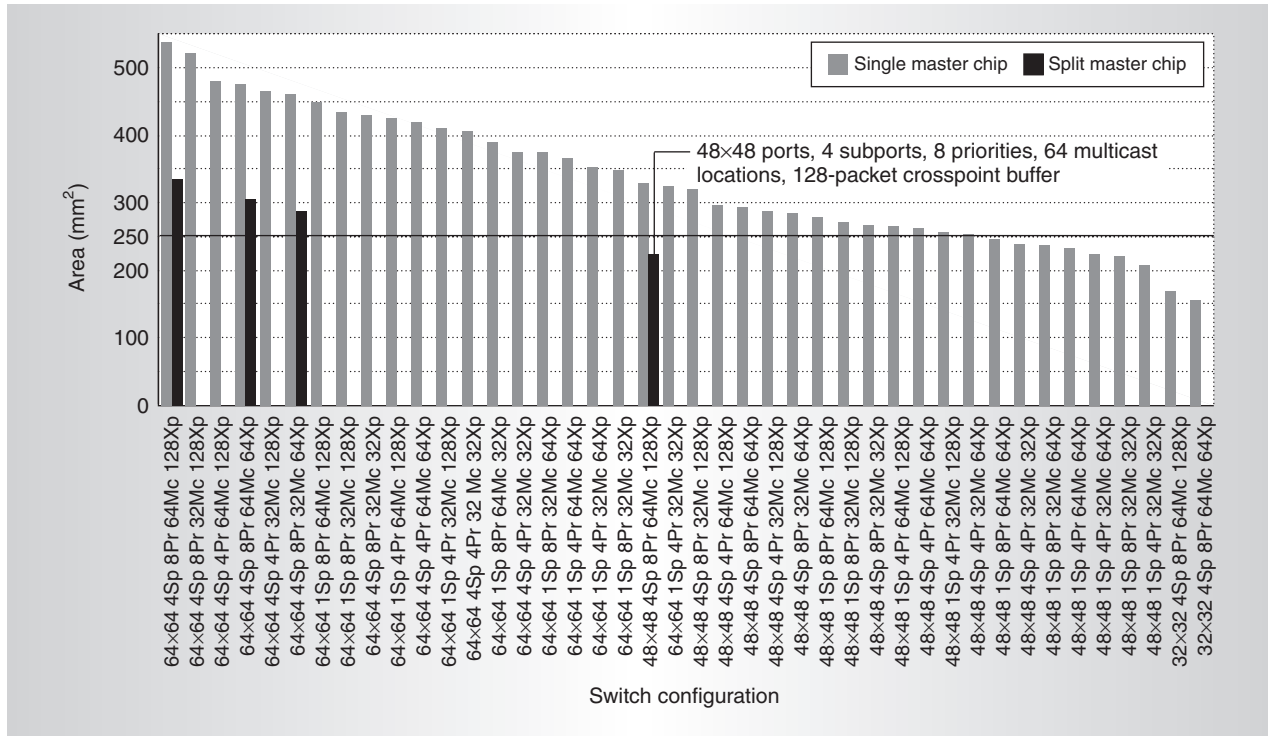


Figure 5. Area use for various single and split master chip configurations.

*Slave chip.* For a crosspoint size of 128 packets, the slave chip fits on a die size of approximately 200 mm<sup>2</sup> with power consumption of 20 W, and pin count of 740 I/O signals. As expected from a buffered crossbar implementation, the greatest area contributor, taking 46 percent of the chip area, is XPM with its 1.25-Mbyte on-chip memory.

*Master chip.* For a 128-packet crosspoint size, a 4-Tbps single-chip implementation of the master would take approximately 540 mm<sup>2</sup> and 56 W. Although buildable, this exceeds our area and power limits. For this chip, the greatest area contributor is XPC, representing 54 percent. Figure 5 shows the area use of multiple configurations with various numbers of ports ( $N \times N$ ), subports (Sp), priorities (Pr), and dedicated multicast locations (Mc), as well as crosspoint buffer sizes (Xp). Given our requirements (4 Sp, 8 Pr, 64 Mc, and 128 Xp) and the area and power limits, Figure 5 shows that a 2-Tbps version of the master chip is feasible (32  $\times$  32 ports).

We considered the high complexity of a 4-Tbps single-chip master less attractive than the following two-step approach. First, based on the master's regular and symmetrical struc-

ture, we design a split master chip and use two identical chips in parallel. This results in a split chip with the same number of I/O pins as an unsplit chip of the same configuration, but only half the queuing structure (XPC), half the Spex interface, and half the switch core control (core). Figure 5 shows sizings for four such split-chip configurations. With this approach, we can build a 48  $\times$  48 (3-Tbps) split master chip in a 0.11-micron CMOS process within the die size and power limits.

Second, to realize the 4-Tbps aggregate throughput, we remap only the master chip to 0.08-micron CMOS to obtain the 64  $\times$  64 configuration. A scaled-down version of the master does not affect the slave chip, and we can operate a 64  $\times$  64 slave with a 48  $\times$  48 master by disabling 16 of the slave's 64 ports.

*Fabric interface and redriver chips.* We do not have room in this brief article to discuss these two chips, but both meet the area, power, and pin requirements.

## Simulated performance

We now evaluate the performance of our proposed 256  $\times$  256 CICQ-based system by

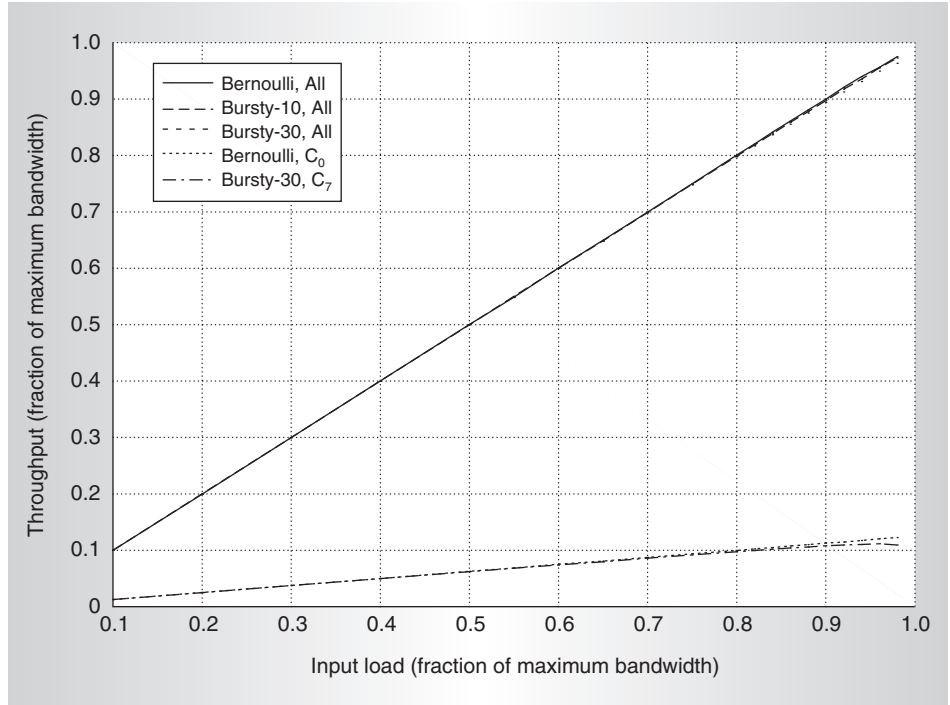


Figure 6. Throughput versus input load with Bernoulli, bursty-10, and bursty-30 traffic.

looking at simulation results. The switch fabric consists of 64 fabric interfaces (those of iFI and eFI), each with four OC-192 external links connecting to the sources/sinks and one OC-768 internal link connecting to the switch core. The switch core is a  $64 \times 64$  buffered crossbar switch, with 64 packet locations per crosspoint, partitioned into four areas of 16 packets for the egress subports. The link round-trip time is 64 packet cycles (at the OC-768 level). The egress buffer on every eFI can store 256 packets per subport, whereas the ingress buffer of the iFI has infinite storage, so no losses occur.

Although we designed the system with a line-rate escalation of 1.6, we simulated it with a speedup of 1. This is because we expect the overhead incurred by the segmentation of variable-length packets into fixed-length packets to cancel the line speedup.

We synthesized traffic consisting of bursty arrivals with geometrically distributed bursts of configurable length, where a burst is a sequence of consecutive packets from one input to the same egress port. The bursts are uniformly distributed over eight classes of service—that is,  $C_p = 12.5$  percent of the offered load for all priorities,  $p \in \{0, 1, \dots, 7\}$ —and the destinations are

uniformly distributed over all 256 egress ports. We simulated the CICQ system described with burst sizes of 1 packet (Bernoulli traffic), 10, and 30 packets, at an average input load ranging from 10 percent to 98 percent.

We can configure the MLQS in various combinations of strict priority and weighted scheduling modes. For this analysis, we configured the MLQS to operate in strict-priority mode—the highest-priority class ( $C_0$ ) always goes first.

We express the end-to-end delays in external packet cycles (at the OC-192 level) and the aggregate throughput as a fraction of the maximum bandwidth. We adjusted the delay values by the constant delay component due to the system's intrinsic transmission latency.

Figure 6 shows the aggregate throughput as a function of the input load (denoted by All) and the throughput of the individual class of service, but only for  $C_0$  Bernoulli and  $C_7$  bursty-30, because all other curves are virtually indistinguishable from these two. The throughput equals the input load in the simulated range up to 98 percent. This demonstrates that even under quite bursty traffic with heavy congestion (high load) the system is blocking free. Several design elements con-

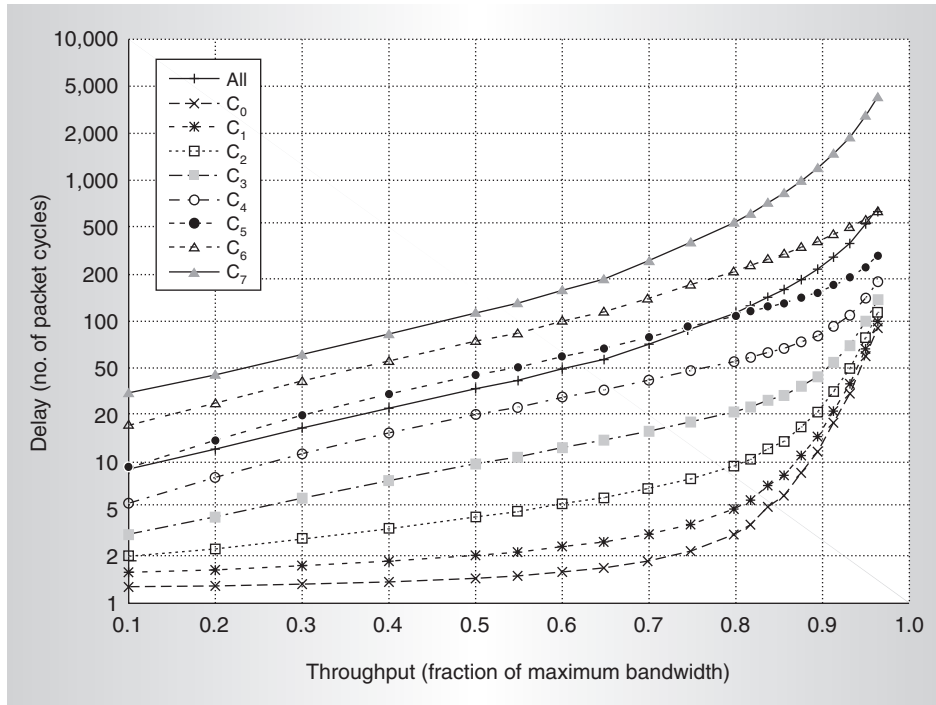


Figure 7. Per-class ( $C_0$  through  $C_7$ ) and aggregate (All) delay for throughput with bursty-30 traffic.

tribute to the elimination of the blocking:

- At the ingress, we used per-destination VOQs (one per output per subport and per class of service).
- The buffered crossbar switch core eliminates output buffer monopolization.
- Dedicated buffers per subport at the egress fabric interface eliminate egress buffer monopolization.
- Systemwide per-destination credit flow control lets flows to uncongested ports proceed while stopping flows to congested ports.

Moreover, the system maintains work conservation under completely unbalanced (hosed) traffic. The crosspoint memory size guarantees this behavior by providing sufficient credits to fully use the available link throughput and therefore avoid output queue underflow and the ensuing loss of throughput.

Figure 7 plots system delays—per class ( $C_0$  through  $C_7$ ) and aggregate (All)—versus aggregate throughput for bursty traffic with an average burst size of 30 packets. The delays are expressed in cycles of 64-byte packets at

the subport level (one packet cycle = 51.2 ns). The delay curves show the class-of-service differentiation and ordering, with the top classes ( $C_0$  through  $C_3$ ) exhibiting significantly lower delay values than the lower-priority classes throughout the load range.

However, the delay for the top classes also increases drastically as the load approaches saturation. This means that the switch core occasionally delays high-priority packets for an exceedingly long time because low-priority packets are stuck in the crosspoint buffers, which can happen because credits (memory locations) are not managed per class of service.

This behavior is unacceptable for the delay- and jitter-sensitive traffic typically carried in the top-priority classes. One way to address this issue is by enhancing the ingress VOQ scheduler with a threshold scheme. A set of thresholds  $\{T_p\}$  correspond to the available credit pool  $CP_{ij}$  for a given input-output pair  $(i, j)$ , with  $0 \leq p \leq 7$ ,  $0 \leq i, j \leq 255$ . The additional condition is that class of service  $C_p$  is only eligible for transmission if  $CP_{ij} > T_p$ . This ensures that any class of service below or equal to priority  $p$  cannot use credits below threshold  $T_p$ .

Because we were interested in the delay dis-

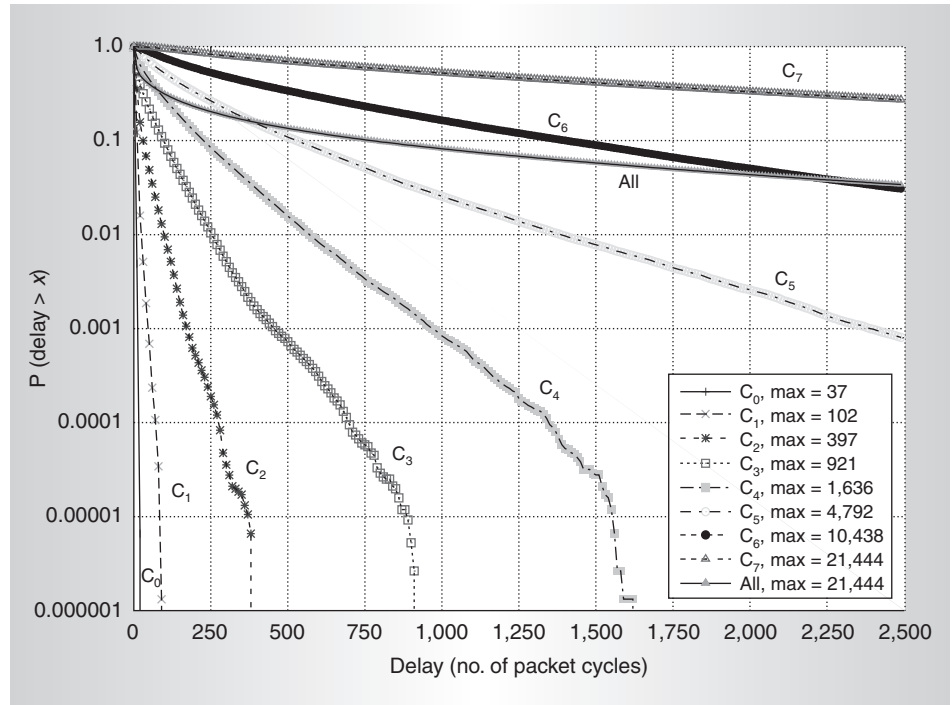


Figure 8. Delay distribution per class with threshold mechanism ( $\Delta T = 4$ ) and bursty-30 traffic.

tribution characteristics of the individual classes of service, we simulated the system we've already described, but with 32 packet locations per crosspoint and per subport (instead of 16). We also added a threshold mechanism with threshold spacing  $\Delta T = 4$ , that is,  $T_p = p \times \Delta T$ . This translates into the system always reserving four credits for  $C_0$ , eight for  $C_0$  through  $C_1$ , 12 for  $C_0$  through  $C_2$ , and so on.

The traffic has an average burst size of 30 packets, destinations are uniformly distributed, and every class of service contributes 12.5 percent of the offered load, which is set to 95 percent of the maximum bandwidth. We sampled the delay values during a simulation run of 100,000 packet cycles and collected the samples in a histogram, which yields the delay distribution graph of Figure 8. This histogram indicates the fraction of packets that have a delay greater than  $x$ , for  $0 \leq x \leq 2,500$  external packet cycles.

Figure 8 shows greatly improved delay behavior: The tails of the high-priority distributions exhibit clear cutoff points. In particular, for  $C_0$  the maximum recorded delay now equals only 37 packet cycles (about 2  $\mu$ s), and its distribution is extremely narrow, because no other class of service can ever block  $C_0$  from

proceeding (four credits are always reserved). This characteristic is also relevant to computer interconnect networks, in which the threshold mechanism guarantees reduced latency even under high load utilization (95 percent).

**I**n conclusion, our CICQ switch architecture combines the scalability of input-buffered switches with the performance characteristics of output-buffered switches. What distinguishes our system is that it supports long cables between the line cards and the switch core without performance degradation under any traffic pattern.

Clearly the link speed and aggregate throughput of this system put it in the realm of optical CIOQ switches. Whereas such optical switches can meet the throughput requirements of communications networks, their end-to-end latency has seldom been considered. However, when one considers latency as the primary performance metric, these systems' cost becomes prohibitive with today's optical technologies.

In contrast, the CMOS-based implementation proposed here excels in both throughput and latency, which qualifies it not only for communications but for computer interconnect

networks as well. This system is also cost-effective by re-using the same modular chipset of four ASICs to build high-performance switches for either of the above-mentioned fields. MICRO

### Acknowledgments

For substantial contributions to this challenging project, we extend special thanks to the IBM Zurich Research Lab's switch team, the Prizma Technology group of IBM La Gaude, France, and the logical and physical design team of IBM Laboratory in Böblingen, Germany.

---

### References

1. R. Rojas-Cessa, E. Oki, and H. Jonathan Chao, "CIXOB-k: Combined Input-Crosspoint-Output Buffered Packet Switch," *Proc. IEEE Globecom 01*, vol. 4, IEEE Press, 2001, pp. 2654-2660.
2. J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks, An Engineering Approach*, Morgan Kaufmann, 2003.
3. C. Minkenberg et al., "Current Issues in Packet Switch Design," HotNets-I 2002—First Workshop on Hot Topics in Networks, to be published in *ACM SigComm Computer Comm. Rev.*, vol. 33, no. 1, Jan. 2003.
4. N. McKeown, "The iSLIP Scheduling Algorithm for Input-Queued Switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, Apr. 1999, pp. 188-201.
5. S-T. Chuang et al., "Matching Output Queuing with a Combined Input Output Queued Switch," *IEEE J. Selected Areas of Comm.*, vol. 17, no. 6, June 1999, pp. 1030-1039.
6. P. Krishna et al., "On the Speedup Required for Work-Conserving Crossbar Switches," *IEEE J. Selected Areas of Comm.*, vol. 17, no. 6, June 1999, pp. 1057-1066.
7. M. Katevenis, D. Serpanos, and E. Spyridakis, "Switching Fabrics with Internal Backpressure Using the ATLAS I Single-chip ATM Switch," *Proc. IEEE Globecom 97*, IEEE Press, 1997, pp. 242-246.
8. F.M. Chiussi, J.G. Kneuer, and V.P. Kumar, "Low-Cost Scalable Switching Solution for Broadband Networking: The ATLANTA Architecture and Chipset," *IEEE Comm. Magazine*, vol. 35, Dec. 1997, pp. 44-53.
9. C. Minkenberg and T. Engbersen, "A Combined Input and Output Queued Packet-switched System Based on PRIZMA Switch-on-a-Chip Technology," *IEEE Comm. Magazine*, vol. 38, Dec. 2000, pp. 70-77.
10. R.P. Luijten et al., "Optimized Architecture and Design of an Output-Queued CMOS Switch Chip," *Proc. 10th Int'l Conf. Computer Communications and Networks*, IEEE CS Press, 2001, pp. 448-453.
11. D.C. Stephens and H. Zhang, "Implementing Distributed Packet Fair Queuing in a Scalable Switch Architecture," *Proc. IEEE Infocom 98*, vol. 1, IEEE Press, 1998, pp. 282-290.
12. M. Nabeshima, "Performance Evaluation of a Combined Input- and Crosspoint-Queued Switch," *IEICE Trans. Communications*, vol. E83-B, no. 3, Mar. 2000, pp. 737-741.
13. T. Javidi, R. Magill, and T. Hrabik, "A High-Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric," *Proc. IEEE Int'l Conf. Communications*, vol. 5, IEEE Press, 2001, pp. 1586-1591.
14. A.K. Gupta, L. Orozco Barbosa, and N.D. Georganas, "16x16 Limited Intermediate Buffer Switch Module for ATM Networks," *Proc. IEEE Globecom 91*, IEEE Press, 1991, pp. 939-943.
15. Y. Doi and N. Yamanaka, "A High-Speed ATM Switch with Input and Cross-Point Buffers," *IEICE Trans. Communications*, vol. E76-B, no. 3, Mar. 1993, pp. 310-314.
16. M. Gusat et al., "Stability Degree of Switches with Finite Buffers under Non-Negligible RTT," *Microprocessor and Microsystems J.*, Elsevier Press, to be published 2003.

**François Abel** is a research staff member at the IBM Zurich Research Laboratory. His research interests include architecture and VLSI design of high-speed switching systems. Abel has a BS in engineering from the École Nationale d'Ingénieurs, Belfort, France, and an MS in electrical engineering from the École Supérieure d'Ingénieurs, Marseille, France. He is a member of the IEEE.

**Cyriel Minkenberg** is a research staff member in the high-speed switching group at the IBM Zurich Research Laboratory. His work and research interests include architecture design, performance modeling, and simulation in the Prizma project. Minkenberg has an MS and PhD in electrical engineering from the Eindhoven University of Technology, the Netherlands.

**NEW FROM IEEE!**

## IEEE INFORMATION TECHNOLOGY LIBRARY (ITeL)

### IEEE Journals, Magazines and Conference Proceedings on:

- Computing
- Communications
- Signal Processing
- Circuits and Systems

An ideal collection for businesses and universities focusing on these technologies!

The IEEE Information Technology Library brings you online access to 39 top-cited IEEE periodicals. Also included are more than 900 IEEE conferences, presenting the very latest technical research.

Developed by the IEEE Computer, Communications, Signal Processing and Circuits and Systems Societies.

**Request a free trial today:**

**+1 800 701 IEEE (4333)**  
(USA/CANADA)

**+1 732 981 0060**  
(WORLDWIDE)

**onlineproducts@ieee.org**  
(EMAIL)



*IEEE Information  
Driving Invention...  
in Information Technology*

**[www.ieee.org/onlinepubs](http://www.ieee.org/onlinepubs)**

**Ronald P. Luijten** is a research staff member at the IBM Zurich Research Laboratory, where he is manager of the high-speed switching team. His research interests include high-speed switching system design, QoS, network processors, and high-speed and high-density electrooptical interconnects. Luijten has an MS in electronic engineering from the Eindhoven University of Technology, the Netherlands. He coorganized the IEEE International Conference on Computer Communications and Networks in 2001 and 2002.

**Mitchell Gusat** is a research staff member at the IBM Zurich Research Laboratory. His research interests include protocols for parallel supercomputing and switching-routing, real-time, QoS, and reconfigurable communication systems, with an emphasis on advanced scheduling and flow control of scalable interconnection networks. Gusat has an MS in electrical engineering from the University of Timisoara, Romania, and an MS in computer engineering from the University of Toronto. He is a member of the IEEE and the ACM.

**Ilias Iliadis** is a research staff member at the IBM Zurich Research Laboratory. His research interests include performance evaluation of computer communication networks, traffic control and engineering for IP and asynchronous transfer mode networks, switch architectures, and stochastic systems. Iliadis has a BS from the National Technical University of Athens, Greece, and an MS and PhD in electrical engineering from Columbia University. He is a member of the Technical Chamber of Greece, IFIP WG 6.3, and Sigma Xi; he is a senior member of the IEEE.

Direct questions and comments about this article to François Abel, IBM Research, Zurich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland; [fab@zurich.ibm.com](mailto:fab@zurich.ibm.com).

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