

A Four-Terabit Single-Stage Packet Switch with Large Round-Trip Time Support

F. Abel, C. Minkenberg, R. P. Luijten, M. Gusat, and I. Iliadis
IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland
fab@zurich.ibm.com

Abstract

We present the architecture and practical VLSI implementation of a 4-Tb/s single-stage switch. It is based on a combined input- and crosspoint-queued structure with virtual output queuing at the ingress, which has the scalability of input-buffered switches and the performance of output-buffered switches. Our system handles the large fabric-internal transmission latency that results from packaging up to 256 line cards into multiple racks. We provide the justification for selecting this architecture and compare it with other current solutions. With an ASIC implementation, we show that a single-stage multi-terabit buffered crossbar approach is viable today.

1. Introduction

We present the design of a multi-terabit single-stage switch fabric that supports from 64 (OC-768) to 256 (OC-192) ports. Owing to the large number of line cards and the size of the switch fabric, the system is distributed over multiple racks, whereby the line cards are packaged tens to hundreds of feet away from the switching core. At OC-768 speed, the impact of the resulting Round-Trip Time (RTT) on the performance of the system architecture is considerable. So far this issue has received insufficient attention in related work. In contrast, the architecture proposed here is designed to directly support a wide range of RTT values in the switching core itself. This constitutes a novel approach in solving the challenges of multi-Tbps switching fabrics.

Because of performance and Class-of-Service (CoS) requirements, we employ a Combined Input and Crosspoint-Queued (CICQ) switch architecture, a subclass of the well-known Combined Input- and Output-Queued (CIOQ) architecture. The CICQ considered consists of a buffered crossbar switch combined with a Virtual Output Queuing (VOQ) arrangement at the ingress. This scheme has been shown to exhibit close to ideal performance characteristics, even under bursty traffic conditions [1].

CICQ has been proposed earlier [2,3], and we will show here that it has become practical because of current advances in CMOS integration density and the addition of VOQ at the ingress. The renewed interest in CICQ is demonstrated in

[4]–[9], without however considering large RTT and feasibility issues.

Our system is built from four different CMOS ASIC building blocks, using a total of 40 chips for the switching fabric and 64 interface chips on the line cards, to achieve 4 Tb/s of aggregate throughput.

The paper is organized as follows: In Section 2 we discuss related switch architectures, and in Section 3 we show our motivations for the proposed architecture and highlight the attractive features of such an arrangement. Section 4 gives a system-level description of the proposed CICQ architecture. Section 5 shows the practical implementation and describes the four-ASIC chipset. Performance simulation results are given in Section 6, and we present our conclusions in Section 7.

2. Related Work

Most current single-stage switch architectures employ VOQ at the ingress, see Fig. 1. The key distinguishing architectural feature is whether scheduling among the ingress VOQs is performed in a *centralized* or *distributed* fashion. The centralized approach typically consists of input buffers organized by destination (VOQ) combined with a bufferless crossbar switch core. Within this category a further distinction can be made between approaches without speed-up (purely input-queued) or with limited speed-up¹ (combined

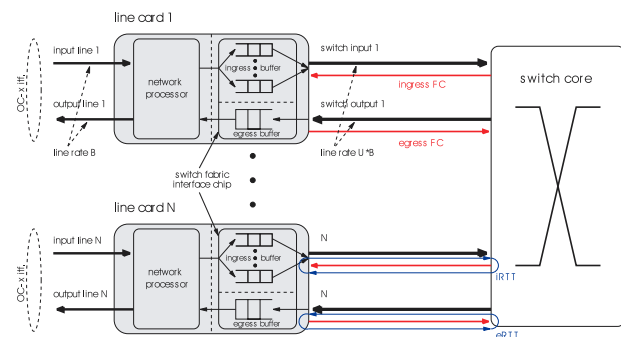


Figure 1. Generic system architecture

¹ As the core is bufferless, this speed-up applies to the ingress buffer read access, the egress buffer write access, the arbitration process, and the entire switch core, including links (see Fig. 1).

input- and output-queued: CIOQ). Both approaches require a centralized arbitration algorithm to resolve input and output contention. In the former case a bipartite graph-matching algorithm such as PIM, iSLIP [10], or similar algorithms, is required, whereas in the latter case even more complex arbitration algorithms [11,12] have been proposed to achieve exact output queuing (OQ) emulation, bringing better QoS support and performance, in particular under non-uniform traffic. The main drawback of this approach is the limited scalability of its centralized arbitration. 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 large port counts because of the super-linear complexity of the arbitration algorithms.

The distributed approach removes the need for centralized arbitration of the inputs by employing a limited number of output buffers with *full speed-up* that are typically integrated in the switch core,² i.e., a CIOQ architecture with an internal speed-up of N . Link-level flow control between the input and output buffers is required to ensure losslessness and prevent output buffer hogging; typically, backpressure, grant, or credit flow control are employed. The main drawback of the distributed CIOQ approach is the complexity of implementing OQ with full speed-up. The traditional implementation is based on a *shared-memory* switch core, where all inputs and outputs simultaneously access a common memory [13]–[15]. However, this implementation does not scale to multi-Tb/s with current technology. Moreover, when using VOQ at the ingress to reduce head-of-line (HoL) blocking, the performance advantage of a shared memory is no longer an argument [16]. This calls for a more distributed output buffer implementation, such as can be achieved by partitioning the shared memory into *dedicated* buffers per groups of inputs and/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. Such a combination of VOQ ingress buffers with a buffered crossbar (CICQ) has recently been proposed in several works [4]–[9], where performance very close to ideal OQ has been shown. The performance depends to a limited degree on the contention resolution mechanisms used at the VOQs and outputs, e.g. RR_RR [6], OCF_OCF [5] and LQF_RR [7].³ However, because of practical implementation issues, the existing proposals have mostly considered only one or a few packets per crosspoint, and paid little attention to the interaction between RTT, work conservation, and QoS scheduling.

3. Motivation

We have adopted the CICQ architecture as the basis for our design. The key reasons for this choice are as follows: first, the CICQ architecture allows contention resolution to be distributed over both inputs and outputs; independent input schedulers resolve input contention, whereas the output buffers resolve output contention to some extent. This results in a simpler, more distributed implementation, as $2N$ schedulers of $O(N)$ complexity are required instead of one $O(N^2)$ scheduler. This leverages newer trends in CMOS technology, i.e. greater density (more parallelism) rather than increased clock speeds (faster logic). Second, most practical, centralized scheduling algorithms are of a heuristic, sub-optimal nature and tend to suffer from performance deterioration under non-uniform traffic conditions [7,8]. It has been demonstrated that CICQ switches exhibit close to ideal performance characteristics even under non-uniform traffic [8].

For any non-blocking CIOQ architecture, whether shared memory or buffered crossbar, it can be shown that the total buffer requirement to support any traffic pattern scales quadratically with N and linearly with RTT: $O(N^2 \times \text{RTT})$. The increasing line speed and the increasing physical size of switch systems imply a large intra-fabric RTT, which, combined with the growing port count, drives the buffer requirements upwards. A shared-memory implementation of such a large buffer does not scale beyond 1 Tb/s throughput, whereas a buffered crossbar implementation is practical, as will be shown in the remainder of this paper.

The CICQ approach inherits a number of additional advantages from the distributed CIOQ approach:

- When there is no contention, packets can proceed immediately to the output without having to wait for a request-grant-accept cycle to complete, thus significantly reducing latency at low utilization.
- 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 as it simplifies the synchronization constraints of large switches that need to be distributed across multiple racks.
- A buffered switch core enables hop-by-hop instead of end-to-end flow control. Even though $N^2 \times \text{RTT}$ buffers are added in the core, the *overall* buffer requirements are reduced because the total egress buffer requirement can be lowered by a factor of $2N$, and the overall minimum buffer requirement drops from $2N^2 \times \text{RTT}$ for an unbuffered core to $(N^2 + N) \times \text{RTT}$ for a buffered core.
- The close-to-ideal performance of this arrangement implies that no external speed-up of the switch core is required. This is key in current Tb/s switches as the major cost and power outlay is in the interconnect.

² Integrating output buffers with fullspeed-up in the switch core (*internal* speed-up) eliminates the need to speed up the links (*external* speed-up).

³ The combination of the scheduler disciplines is denoted XX_YY, where XX denotes the VOQ arbitration and YY the OQ arbitration. These can be for example Round Robin (RR), Oldest Cell First (OCF), or Longest Queue First (LQF).

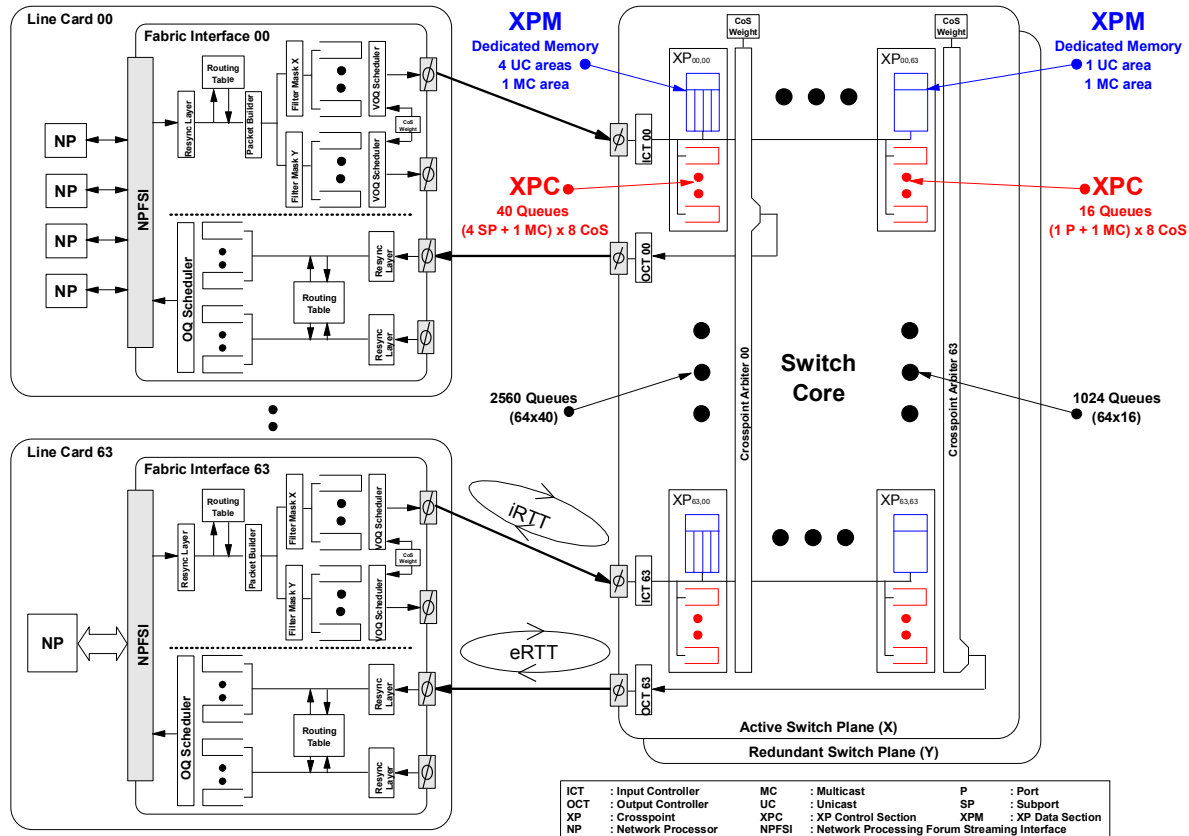


Figure 2. CICQ switch fabric – logical view

- The crosspoint memories can be operated at the line rate, a major reduction compared to shared-memory architectures.
- A buffered crossbar is inherently free of *buffer hogging*, which can cause serious performance degradation in shared-memory switches that do not provide efficient congestion-control mechanisms [1].
- Distributed control of buffering enables flow decoupling, which improves fairness [4,17].

4. Distributed Packet Routing Switch Architecture

A conceptual block diagram of the proposed architecture is shown in Fig. 2, and we will refer to it as the distributed packet routing switch architecture.

4.1. System Architecture Requirements and Overview

Our switch fabric supports 64 full-duplex physical ports each of which is configured as either one OC-768 (40 Gb/s) interface or four OC-192 (10 Gb/s) interfaces, referred to as sub-ports (SP), or any combination thereof. The switch fabric supports eight classes of service by means of priorities, e.g.

for IEEE 802.1D/Q support, and provides flexible distributed schedulers for service differentiation and QoS delivery. The contention resolution is round robin (RR) for the input VOQs and weighted round-robin (WRR) for the crosspoint arbitration [17]. Service differentiation is provided by a so-called multi-layer QoS scheduler (MLQS) at each of the three points of contention: ingress fabric interface, switching core, and egress fabric interface. MLQS is a combination of strict priority and weighted scheduling; the latter guarantees a minimum bandwidth but does not preclude using the maximum thereof. With this scheme, each CoS is mapped into either a strict or weighted scheduling group. This allows some classes of service with low and strict delay jitter requirements to be handled by the strict priority scheduling. Classes of service requiring minimum guaranteed bandwidth but no specific delay bound are mapped into the weighted scheduling group.

For lossless operation we employ a reliable link-level credit-based flow control between port cards and switching core. The system is guaranteed to be non-blocking at the port, sub-port, and CoS levels.

Output ports are addressed by a direct bitmap, which inherently supports multicast. Sub-ports are addressed by an additional 2-bit sub-port destination (SPD) identifier. To reduce overhead, we define four 16-port groups (O[00:15],

O[16:31], O[32:47], 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. To perform a multicast, up to four packets can be required. The fabric interface (FI) is implemented in a combined ingress-egress chip that connects to any Network Processor (NP) unit that is compliant with the NPF Streaming Interface (NPF SI). The data stream entering the switch fabric is segmented into fixed-sized packets of either 64 or 80 bytes. Although the system achieves 100% throughput without requiring speedup [8], it is still operated with a line rate escalation of $U = 1.6$ (i.e. 64 instead of 40 Gb/s) to compensate for the switch packet header and segmentation overhead (see Fig. 1). This results in a switch fabric with aggregate throughput of 4 Tb/s (2.5 Tb/s at the OC-768 and/or OC-192 levels).

A chipset of four ASIC devices implements the switch architecture, three for the switch core and one for the fabric interface. This chipset can be used to scale from 1 to 4 Tb/s, by employing 27 to 104 devices, respectively.

4.2. Ingress Fabric Interface

In our architecture, the VOQ is organized per destination and per CoS. The ingress path of the fabric interface (iFI) supports 256 unicast destinations and four multicast group destinations (one per BME). Including CoS, this leads to a total of $(256 + 4) \times 8 = 2080$ VOQs. 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 4096 incoming packets and performs lossless switch-over between active and redundant planes. The VOQ scheduler is based on MLQS and RR arbitration. The RR algorithm is used to select packets of the same CoS when multiple queues are eligible for transmission.

4.3. Switch Core

The switching core (SC) is based on a crosspoint queuing architecture with additional QoS support. It consists of a single-stage switch architecture with 64x64 logical links operated at 64 Gb/s. Each logical link is implemented with 32 serial links operated at 2.5 Gb/s and 8b/10b encoding. At 64 Gb/s, a 64-byte packet lasts 8 ns. The sub-port configuration is realized by multiplexing the four sub-port streams onto a single logical link.

The internal memory and queuing structure is realized by a 64x256 arrangement to cover all cases, including the one where all 64 destination ports consist of four sub-ports.

Logical crosspoint architecture description: We define $XP_{i,o}$ (Fig. 2) to be the logical crosspoint dedicated to physical input i and physical output o . There are a total $64 \times 64 = 4096$ XPs, each having a separate data and control path section. Organization of the data and control sections depends on the output port configuration, i.e., with or without sub-port support. For example in Fig. 2, crosspoints corresponding to output 0 are configured in sub-port mode, whereas those corresponding to output 63 are not.

We define $XPM_{i,o}$ (Fig. 2) to be the data section of a logical crosspoint. It implements a crosspoint memory buffer dedicated to physical input i and physical output o , which consists of a unicast (UC) dedicated area and a multicast (MC) dedicated area. If physical output o is operated with sub-port support, the UC area is further partitioned into four dedicated areas per sub-port (e.g. $XP_{0,0}$ in Fig. 2), required to avoid buffer hogging at the sub-port level. The storage address for a UC packet arriving at $XPM_{i,o}$ is provided by the crosspoint control section. Each dedicated UC area represents an individual flow control domain that the control section manages by means of specific credits. There are a maximum of $64 \times 64 \times 4 = 16K$ UC flow-control domains.

We define $XPC_{i,o}$ (Fig. 2) to be the control section of a logical crosspoint. It 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 UC area(s) of the data section, and a credit manager that flow controls the UC dedicated area(s). This structure is a set of 40 output queues corresponding to four sub-ports queues plus one multicast queue times eight CoS, which is necessary to guarantee non-blocking behavior at the sub-port level, while CoS queuing is required for QoS scheduling at the current output contention point. If physical output o is operated without sub-port (e.g. $XP_{0,63}$), then only 16 out of 40 queues are used. The switching core implements a total of $40 \times 4K = 160K$ queues.

XPM dimensioning: Considering that switches and routers are typically housed in 19–21-inch shelves, each having approx. 16 line cards, the proposed architecture implies a minimum of 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 switch-fabric- internal RTT. We consider this an important new design factor that so far has not been addressed extensively in current CICQ proposals. We define the input Round-Trip Time (iRTT) as the composite of the transmission packet delay from an ingress fabric interface to a XP destination plus the transmission of the flow control information back to the iFI (see Fig. 1). iRTT is expressed in number of packets being in flight, and includes the contribution of the cables, the logic at the receiver and transmitter sides, such as data deskewing and alignment, plus the VOQ and XP arbitration times. eRTT is defined similarly at the egress side. Credit flow control inherently satisfies the losslessness property, but to satisfy the input work-conservingness property under any traffic scenario, the size of the $XPM_{i,o}$ must be scaled proportionally to iRTT ($XPM_{i,o} \geq iRTT$), which translates into at least iRTT credits being available to fully utilize the link (memory size and credits are linear functions of RTT).

Our requirement is to support any traffic pattern over a distance of 100 feet of interconnect between the FI and the SC, which translates into $iRTT = eRTT = 64$ packets. With this number of credits (64) one can operate over interconnect lengths of more than 100 feet if the range of traffic patterns

is limited. For example, considering only uniformly distributed traffic allows us to operate with link lengths of up to 10,000 feet and still achieve 100% throughput. Finally, to fully decouple unicast and multicast performance we allocate $2 \times iRTT$ packets per XPM, where the size of the MC area is programmable between 0 and $iRTT$ packets.

Unicast and multicast operation: Figure 2 shows the distributed output buffers and output queues dedicated to a particular input port. In this design, arriving packets are broadcast on a unidirectional bus that connects to all XPs of that specific input.

For unicast packets, every XP in a row filters out the bitmap and the BME fields of the header to determine if a packet is destined to the output it connects to. If yes, a UC store address is internally generated by the XPC and used to write the packet into the XPM. At the same time, the address is also enqueued in the XPC according to the SPD and the CoS fields.

This broadcast-and-select behavior also provides multicast capability, as multiple bits set into the bitmap will select multiple XPs. Contrary to UC, the MC store address is generated by an MC address manager (MCAM) common to 16 output ports (one MC group), and is broadcast along with the packet. This guarantees that all the replicated copies of an MC packet get written at the same address within all the dedicated MC areas, and implies that an MC group constitutes one flow control domain. The use of a unique MC address simplifies the design of the MCAM which generates and recycles this type of addresses. The XPC has one specific MC queue per CoS.

Crosspoint arbitration description: There is one crosspoint arbiter per output port. Every 8 ns, the arbiter selects one out of the 64×40 queues. The algorithm is as follows:

- 1) Mask out those destinations that have either no traffic or no credit available for the corresponding sub-port or MC destination.
- 2) Select a sub-port or MC destination according to the following RR pattern: $[SP_0, MC, SP_1, MC, SP_2, MC, SP_3, MC]$.
- 3) For the selected sub-port or MC destination, select a CoS according to MLQS.
- 4) Choose a crosspoint corresponding to the selected CoS and destination according to a WRR schedule.

Steps 2) and 4) are of spatial nature, while 3) is temporal.

4.4. Egress Fabric Interface

Owing to the reduced memory requirement, which is driven only by the link-level flow control ($eRTT = 64$), the egress fabric interface (eFI) can be implemented in the same chip as the ingress (Figs. 1, 2). If sub-ports are enabled the egress buffer is partitioned into five areas (one per sub-port and one for MC), otherwise into two areas (one for UC and one for MC). This scheme is replicated per plane (X and Y).

Queuing is performed per switch plane, sub-port destination and CoS ($2 \times 4 \times 8 = 64$ queues). Multicast at port level is handled by the switch core; MC at sub-port level is expanded by the egress adapter by means of a routing index that addresses a multicast table. At the NPFSI egress side, packet scheduling is also based on MLQS arbitration.

5. VLSI Implementation

In this section we demonstrate the feasibility of the proposed CICQ system and its implementation into a chipset of four different ASICs. Because of space limitation we mainly focus on the switch core, which is the most challenging part.

5.1. Cost and Power Requirements

To remain cost-effective, we limit the die size to a maximum of 250 mm^2 and the pin count to approx. 1000 signal IOs (total ≈ 1500 pins package). For design-time reasons, only standard cell design methodologies, processes and packages can be considered.

To comply with stringent Network Equipment Building Standards (NEBS) rules, a single board must not dissipate more than 250 W (some applications require even less than 150 W). At the same time, single-chip power dissipation must remain under 25 W to avoid hot spots and remain cooled by forced air.

5.2. Sizing Assumptions

The sizing and design assumptions are based solely on currently available, proven technologies. The target CMOS technology is a $0.11\text{-}\mu\text{m}$ copper technology (Cu-11). The sizing was performed with the following assumptions: all critical parts of the design have been VHDL-coded and synthesized. This includes a specific 4×4 cluster that implements the control section of 16 logical XPs (4 inputs \times 4 outputs), an equivalent 4×4 cluster for the data-section implementation, and the output-queue scheduler that performs crosspoint arbitration and MLQS. Sizing numbers for the digital and analog hard macros, such as memories and serializers/deserializers (SERDES), are retrieved from the standard cell ASIC library. All other chiplets were individually estimated based on previous switch designs. A wiring factor of 100% is taken into account for all control logic except for the hard macros which are already placed and routed.

5.3. Physical Implementation

In addition to the quadratic memory challenge, a buffered crossbar is also limited by IO pins and power requirements.⁴ Given the high-speed and high-density SERDES available in ASIC libraries today⁵ ($\approx 125 \text{ mW}$ per 2.5 to 3.2 Gb/s full-duplex), a 4-Tb/s switch requires about 10,000 pins and

⁴ This IO limitation is inherent in any crossbar, whether buffered or bufferless.

⁵ Fujitsu, IBM, LSI, Lucent, TI.

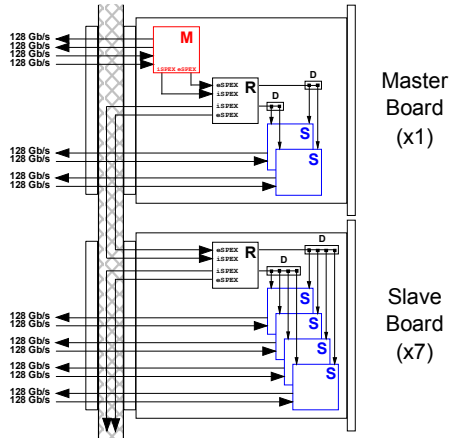


Figure 3. Physical switch core implementation

dissipates roughly 256 W of IO power, thus requiring at least 16 chips to meet the pin and power constraints.

To overcome the inherent IO and memory limitations of buffered crossbars, we expand the port rate and buffering capacity by stacking multiple (31) switch slices operated in parallel and have them controlled by a single master chip. This technique of parallel sliced switching has been successfully used to scale shared-memory architectures up to 1 Tb/s [15,16], and is often referred to as port speed expansion (SPEX) mode or distributed switch architecture with centralized control. Note that this architecture is based on a single switch domain and therefore does not provide multiple paths. However, the proposed 4-Tb/s switch core can be used as a building block with external load-balancing controllers to realize an $N \times N$ multipath switch with higher external port rate.

Figure 3 shows the proposed implementation of the switching core based on the SPEX concept distributed over 8 physical boards. It consists of one master chip (M) and 30 slave chips (S), organized in a combination of a chain- and tree-based topology. The master implements the control section XPC of the 4096 XPs, while each of the 30 slaves implements 1/30 of the data section XPM.

Incoming packets are partitioned by the iFI into $k = 32$ segments of 16 bits (or 20 bits if packet size is 80 B) before being sent to the switching core over k different links, each operating at 2.5 Gb/s. The first two segments containing the packet header are sent to the master module, whereas the $k-2$ other segments containing only data payload are transmitted 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 carried by the header. At the same time, a 2-byte control information comprising the destination XPM within a row, the store address within that XPM, and a UC/MC flag, is transmitted to the slaves over the ingress speed expansion interface (iSPEX). This information directs the slaves how to handle their data segments. Another reason why MC packets share the same store address is to reduce bandwidth of the iSPEX

interface. Every time a packet address is scheduled for transmission in the master, the corresponding data segments need to be retrieved 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. Note that the content of the header is not stored in the master chip but is rebuilt at transmission time: the CoS and SPD fields are retrieved accordingly from the read queue, while the bitmap and BME fields are replaced with inband flow control information.

There is one re-driver chip (R) per card to reshape the iSPEX and eSPEX interface signals (also implemented with high-speed SERDES) before forwarding them via a backplane to the next daisy-chained card. A driving without reshaping of the signals is performed at board level by a commercially available driver (D).

The 4-Tb/s switch core is packaged over eight physical cards, each consuming a single slot of a 19–21-inch shelf. Every card connector handles a bisectonal bandwidth of 1 Tb/s (512 Gb/s in + 512 Gb/s out), which translates into a minimum card edge of 18 inches, given the state-of-the-art 2.5-Gb/s connectors and plug-compatible cable assemblies (2 mm VHDM-HSD connectors offer 36 differential pairs per linear inch). Connecting eight times 1 Tb/s over a backplane requires approx. 4000 differential wire pairs and is only feasible by keeping the wires short and spatially localized. This is achieved by plugging in the eight switch core blades only every other slots (e.g., slots 0, 2, 4, ..., 14), thus always leaving an entire slot free for the interconnect between line cards and switch core.

- If the interconnection links between line cards and switch core use copper cables⁶, the cables directly plug into the rear side of the back plane (Fig. 3) at specific empty slots (i.e., slots 1, 3, 5, ..., 15). The cable assemblies use the same signal distribution and connector type as the switch core blade, thus enabling short (one inch) and direct one-to-one back-plane wiring.
- If the interconnect links use optical fibers, eight additional optical blades are required to perform the electrical-to-optical adaptation and vice-versa. For the same wiring reasons, these blades are interleaved with the switch core blades, and plugged into the empty slots 1, 3, 5, ..., 15.

5.4. Master and Slave Chip Sizing

Figure 4 shows the area utilization and power consumption of the master and slave chips. The x -axis indicates four logical crosspoint sizes, which correspond to $RTT/4$, $RTT/2$, RTT and $2 \times RTT$ packets. The seven parts of the stacked bar represent the contribution of the SERDES IO interface (TX and RX), the SERDES SPEX interface (TX or RX), the logical crosspoints (XPM or XPC), some additional control for the switch core (CORE), for the switch IO protocol

⁶ The considered SERDES can drive 15 feet of coaxial cable and two connectors, directly from ASIC pin to ASIC pin.

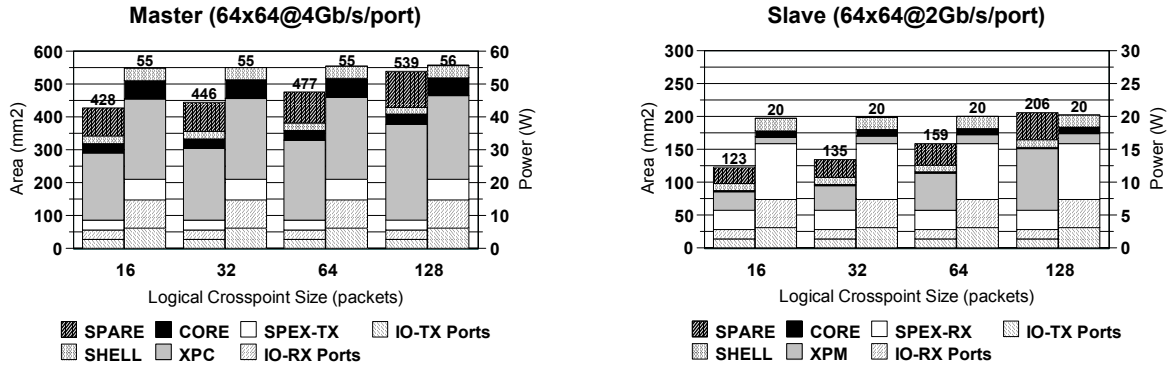


Figure 4. Master and slave chip sizing

(SHELL), and some spare area for the global wiring (SPARE).

Slave chip (XP size of 128): With a die size of approx. 200 mm², a power consumption of 20 W and a pin count of 740 IO signals, the slave chip is clearly feasible. As expected from a buffered crossbar implementation, the greatest area contributor is XPM (46%) with its 1.25 MB of on-chip memory. A well-known characteristic of distributed architectures with centralized control is the large amount of information flowing from the master to the slaves. This typical drawback also hits the proposed implementation, particularly on the power consumption side, where the control-section bandwidth (SPEX-RX) dissipates as much as the entire data-section bandwidth (IO-RX and IO-TX Ports).

Master chip (XP size of 128): A 4-Tb/s single-chip implementation of the master, with approx. 540 mm² and 56 W, exceeds our area and power limits, although it could be built. In this case the greatest area contributor is XPC, which represents 54%. Figure 5 shows the area utilization of

multiple configurations with varying number of ports ($N \times N$), of sub-ports (Sp), of priorities (Pr) and of dedicated MC locations (Mc) as well as crosspoint buffer sizes (Xp). Given our requirements (4Sp 8Pr 64Mc 128Xp) and the area and power limits, Fig. 5 shows that a 2-Tb/s version of the master chip is feasible (32x32).

We considered the high complexity of a 4-Tb/s single-chip master less attractive than the following two-step approach. First, based on the regular and symmetrical structure of the master, we design a split master chip and use two identical chips in parallel. This results in a split chip that has the same number of IO pins as a unsplit chip of the same configuration, but with only half of the queuing structure (XPC), half of the speed expansion interface (SPEX), and half of the switch core control (CORE). Four chip configurations have been sized and are shown in Fig. 5. With this approach, a 48x48 (3 Tb/s) split master chip can be built in 0.11- μ m CMOS process within the die size and power limits. Second, to realize the 4 Tb/s of aggregate throughput we will

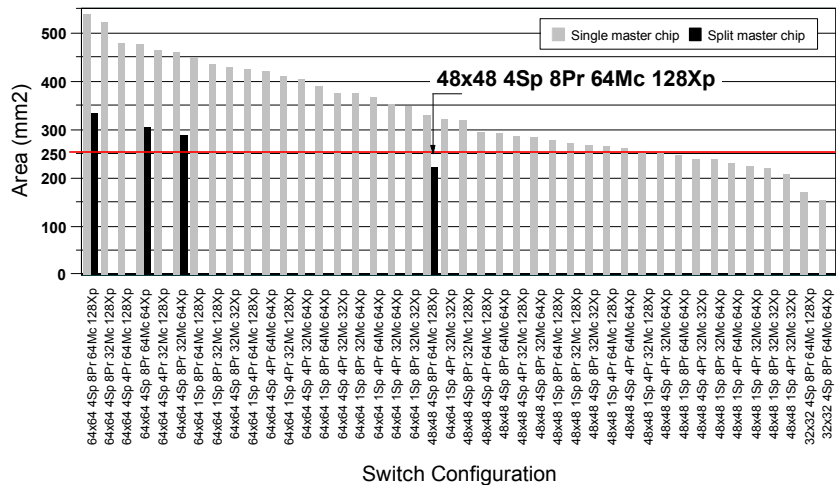


Figure 5. Area utilization for various single and split master chip configurations.

re-map only the master chip to 0.08- μ m CMOS to obtain the 64x64 configuration. Note that a scaled-down version of the master does not impact the slave chip, and that a 64x64 slave can be operated with a 48x48 master by disabling 16 of the 64 ports.

Adapter and re-driver chips: For space reasons, these two chips are not discussed, but both meet the area, power, and pin requirements.

6. Simulated Performance

In this section we evaluate the performance of the proposed system by means of simulation. We have simulated a 64x64 system (16 ports, 4 sub-ports) with 8 classes of service and an RTT of 64 packet cycles between line cards and switch core. Every crosspoint memory has 128 ($2 \times \text{RTT}$) packet locations, i.e., 32 per sub-port. The egress fabric interface of the line card has 1024 packet locations, 256 per sub-port, whereas the ingress side has infinite buffer space, so that no losses occur.

Although the system is designed with a line-rate escalation $U = 1.6$, we have simulated it with a speedup of one because we expect this line speedup to be canceled by the packet header and segmentation overhead.

The traffic is synthetic, consisting of bursty arrivals with geometrically distributed burst sizes, where a burst is a sequence of consecutive packets from one input to the same output (average burst size = 30 packets). The bursts are uniformly distributed over the 8 available CoS, i.e. $C_i = 12.5\%$ of the offered load for all $i \in \{0,1,\dots,7\}$, and the destinations are uniformly distributed over all outputs.

The multi-layer QoS schedulers can be configured in various combinations of strict priority and/or weighted scheduling modes. Here we study the case in which the multi-layer QoS schedulers are configured to operate in strict priority mode, i.e. the highest-priority class (C_0) always goes first.

Figure 6 shows the throughput as a function of the input load, whereas Fig. 7 shows the overall system delay as a function of the input load, each figure containing 9 curves: one per CoS (C_0 - C_7) plus one aggregate curve for all traffic (ALL). The x-axis shows the overall input load as a fraction of the maximum bandwidth. The y-axis in Fig. 6 shows the throughput, also as a fraction of the maximum bandwidth, whereas the y-axis in Fig. 7 shows the average packet delay expressed in packet cycles of the switch core. Note that the packet cycle of the switch core is four times shorter than that of the external sub-ports (OC-768 versus OC-192).

The throughput equals the input load up to the simulated maximum load of 98%. This demonstrates that even under quite bursty traffic with heavy congestion (high load) the system is free of HoL blocking. The system has been rigorously designed to eliminate HoL blocking, because

1) per-destination VOQs (one per output per sub-port and per priority) are used at the ingress,

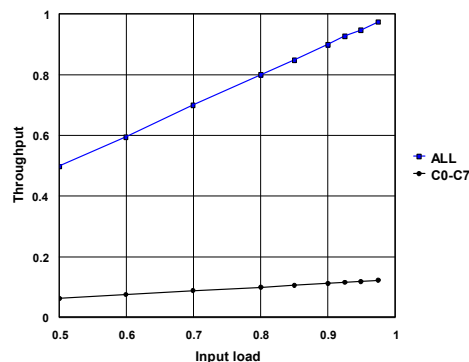


Figure 6. Throughput versus input load

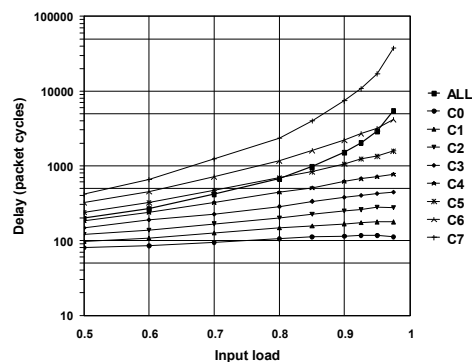


Figure 7. System delay versus input load

- 2) the buffered crossbar switch core eliminates output buffer hogging,
- 3) dedicated buffers per sub-port at the egress fabric interface eliminate egress buffer hogging, and
- 4) per-destination hop-by-hop credit flow control allows flows to non-congested ports to proceed while flows to congested ports are stopped.

Moreover, the system remains work-conserving under completely unbalanced, i.e. totally directional, traffic [6,8]. This behavior is guaranteed by the crosspoint memory size, which provides sufficient credits to fully utilize the available link throughput and therefore avoid OQ underflow and the ensuing loss of throughput.

The delay curves clearly show the CoS differentiation and ordering, with the top classes (C_0 - C_3) being largely unaffected by the load increase, whereas the delay for the lower-priority classes increases drastically. Figures 8-10 show the delay in the ingress fabric interface buffers, in the switch core, and in the egress fabric interface buffers, respectively. We observe that at loads up to 85% the majority of the delay occurs in the eFI buffers, whereas at higher loads, as the eFI buffers saturate, the distribution shifts towards the switch and then, as the switch buffers also saturate, towards the iFI. Figure 11 illustrates this and shows the relative delay contributions of the iFI, switch core, and eFI for all priority classes together as a function of input load.

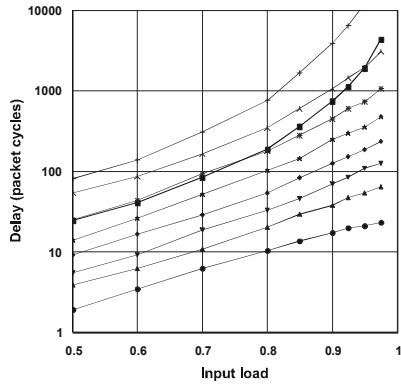


Figure 8. Ingress delay versus load

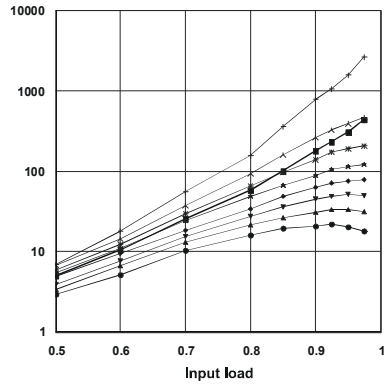


Figure 9. Switch core delay versus load

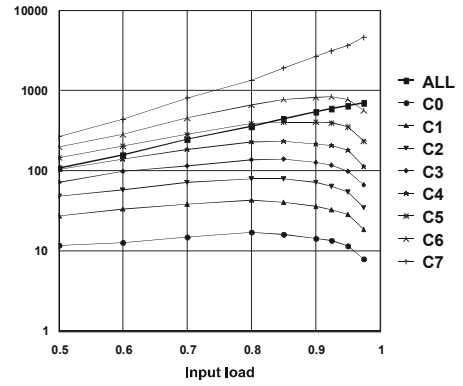


Figure 10. Egress delay versus load

Figure 10 exhibits a non-intuitive behavior: the average latency experienced in the eFI buffer (and to a lesser degree in the switch) of all but the lowest-priority-class (C_7) packets decreases as the load increases beyond a certain point (80%–90%, depending on the priority class), although the overall system packet latency still increases. This behavior results from a trade-off between sub-port fairness and CoS scheduling. An in-depth discussion on this behavior would exceed the scope of this paper.

7. Conclusions

We have presented a multi-terabit single-stage distributed packet-routing switch architecture which combines the scalability of input-buffered switches with the performance characteristics of output-buffered switches.

Based on this architecture, we have presented the design of a 4-Tb/s switch fabric with up to 256 ports, and shown the practical implementation of this system using today's state-of-the-art CMOS technology. What distinguishes our system is that it supports cables of up to 100 feet between the line cards and the switch core, without performance degradation under *any* traffic pattern. Such distances result in large round-trip times, and arise from the necessity to distribute multi-terabit systems over multiple shelves and racks.

At the time of writing, this architecture is finalized, simulated, synthesized (major blocks), and ready to be implemented.

Acknowledgments

The authors extend their special thanks to the IBM Zurich Research Lab's switch team as well as the PRIZMA Technology group of IBM La Gaude, France, and the logical and physical design team of the IBM Laboratory in Böblingen, Germany, for their substantial contributions to this challenging project.

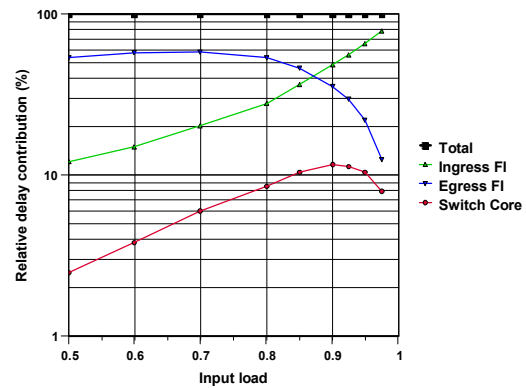


Figure 11. Relative delay composition (ALL)

References

- [1] J.W. Causey and H.S. Kim, "Comparison of Buffer Allocation Schemes in ATM Switches: Complete Sharing, Partial Sharing, and Dedicated Allocation", *Proc. ICC '94*, pp. 1164-1168.
- [2] A.K. Gupta, L. Orozco Barbosa, and N.D. Georganas, "16x16 Limited Intermediate Buffer Switch Module For ATM Networks", *Proc. GLOBECOM '91*, pp. 939-943.
- [3] Y. Doi and N. Yamanaka, "A High-Speed ATM Switch with Input and Cross-Point Buffers", *IEICE Trans. Commun.*, vol. E76-B, no. 3, March 1993, pp. 310-314.
- [4] D.C. Stephens and H. Zhang, "Implementing Distributed Packet Fair Queueing in a Scalable Switch Architecture", *Proc. IEEE INFOCOM '98*, San Francisco, CA, vol. 1, pp. 282-290.
- [5] M. Nabeshima, "Performance Evaluation of a Combined Input- and Crosspoint-Queued Switch", *IEICE Trans. Commun.*, vol. E83-B, no. 3, March 2000, pp. 737-741.
- [6] R. Rojas-Cessa, E. Oki, Z. Jing, and H. Jonathan. Chao, "CIXB-1: Combined Input-One-cell-Crosspoint Buffered Switch", *Proc. 2001 IEEE Workshop on High Performance Switching and Routing*, Dallas, TX, May 2001, pp. 324-329

- [7] T. Javidi, R. Magill, and T.Hrabik, "A High-Throughput Scheduling Algorithm for a Buffered Crossbar Switch Fabric", *Proc. ICC 2001*, Helsinki, Finland, June 2001, vol. 5, pp. 1586-1591.
- [8] R. Rojas-Cessa, E. Oki, and H. Jonathan. Chao, "CIXOB-k : Combined Input-Crosspoint-Output Buffered Packet Switch", *Proc. GLOBECOM '01*, vol. 4, pp. 2654-2660.
- [9] K. Yoshigoe and K.J. Christensen, "A Parallel-Polled Virtual Output Queued Switch with a Buffered Crossbar", *Proc. 2001 IEEE Workshop on High Performance Switching and Routing*, Dallas, TX, May 2001, pp. 271-275.
- [10] N. McKeown, "The iSLIP Scheduling Algorithm for Input-queued Switches", *IEEE/ACM Trans. Networking*, vol. 7, no. 2, April 1999, pp. 188-201
- [11] S-T. Chuang, A. Goel, N. McKeown, and B. Prabhakar, "Matching Output Queueing with a Combined Input Output Queued Switch", *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, June 1999, pp. 1030-1039.
- [12] P. Krishna, N. Patel, A. Charny, and R.J. Simcoe, "On the Speedup Required for Work-conserving Crossbar Switches", *IEEE J. Sel. Areas Commun.*, vol. 17, no. 6, 1999, pp. 1057-1066.
- [13] M. Katevenis, D. Serpanos, and E. Spyridakis, "Switching Fabrics with Internal Backpressure Using the ATLAS I Single-chip ATM Switch", *Proc. GLOBECOM '97*, Phoenix, AZ, Nov. 1997, pp. 242-246.
- [14] F.M. Chiussi, J.G. Kneuer, and V.P. Kumar, "Low-cost Scalable Switching Solution for Broadband Networking: The ATLANTA Architecture and Chipset", *IEEE Commun. Mag.*, vol. 35, Dec. 1997, pp. 44-53.
- [15] C. Minkenberg and T. Engbersen, "A Combined Input and Output Queued Packet-switched System Based on PRIZMA Switch-on-a-Chip Technology", *IEEE Commun. Mag.*, vol. 38, Dec. 2000, pp. 70-77.
- [16] R.P. Luijten, F. Abel, M. Gusat, and C. Minkenberg, "Optimized Architecture and Design of an Output-Queued CMOS Switch Chip", *Proc. 10th Int'l Conf. on Computer Communications and Networks*, Scottsdale, AZ, Oct. 2001, pp. 448-453.
- [17] M.G.H. Katevenis, "Fast Switching and Fair Control of Congested Flow in Broadband Networks", *IEEE J. Sel. Areas Commun.*, vol. 5, no. 8, Oct. 1987, pp. 1315-1326.