

Control Path Implementation for a Low-Latency Optical HPC Switch

Cyriel Minkenbergh, François Abel, Peter Müller,
Raj Krishnamurthy, Mitchell Gusat
IBM Research GmbH, 8803 Rüschlikon, Switzerland

B. Roe Hemenway
Corning Incorporated, Science and Technology
Corning, New York 14831

Abstract—A crucial part of any high-performance computing system is its interconnection network. In the OSMOSIS project, Corning and IBM are jointly developing a demonstrator interconnect based on optical cell switching with electronic control. Starting from the core set of requirements, we present the system design rationale and show how it impacts the practical implementation. Our focus is on solving the technical issues related to the electronic control path, and we show that it is feasible at the targeted design point.

I. INTRODUCTION

A. Motivation

Massively parallel high-performance computing (HPC) systems require large-scale, high-bandwidth, low-latency interconnection networks (ICTNs). The Optical Shared Memory Supercomputer Interconnect System (OSMOSIS) project [1] explores the opportunity to promote the role of optical switching technologies in such systems. As high-end HPC systems (supercomputers) are typically highly distributed and very large, having diameters that can span tens to hundreds of meters, the capability to cover such distances with low power is a significant advantage. An optical switch also typically dissipates significantly less power than a comparable electronic switch. Given that chip design today is becoming increasingly constrained by the ability to power and cool the packages, this is a key advantage. Also, the low sensitivity to EMI is a clear benefit, especially when covering such distances. The main disadvantage still is the cost of optical components.

The objectives of the OSMOSIS project are twofold: First, we aim to solve the technical challenges involved in *building a demonstrator* ICTN that meets a specific set of ambitious target requirements (Sec. I-B), using components and technologies currently available. Second, we aim to accelerate the cost reduction of all-optical switches for HPC. This paper primarily addresses the technical issues encountered in the electronic (control path) domain. The cost reduction aspect involves achieving denser optical integration (part count reduction) and finding—in addition to the low-volume HPC market—a high-volume market for these optical components.

First, we review the requirements in Sec. I-B. Based on these requirements, we provide a step-by-step derivation of the key system design choices in Sec. II, with a focus on the control path. This leads to the OSMOSIS system architecture, briefly reviewed in Sec. III. We discuss the challenges posed

by the design and implementation of the control path and our solutions in Sec. IV. We demonstrate how we map this design to a distributed, FPGA-only implementation, and we show that this implementation is feasible and can be expected to meet the requirements. Finally, we conclude in Sec. V.

B. Requirements

HPC interconnects must be designed to handle the arrival, buffering, routing, forwarding, and error management of high-rate data and control streams in scalable local networks. HPC switches must deliver very low latency, an extremely low bit-error rate, a high data rate, and extreme scalability. Moreover, they must be efficient for bursty traffic, with demands ranging from very small messages (e.g., collectives and syncs) to bulk dataset transfers (e.g., memory pages).

The specific requirements for our demonstrator system include less than 1 μ s latency measured application–application, switch ports operating at 40 Gbit/s with 75% available for user payload, a 10^{-21} bit-error rate, and an optical data path. Furthermore, the system must be designed to scale to at least 2048 nodes, and must be capable of scaling to higher line rates (at least 160 Gb/s). Finally, all electronic logic must be implemented using only FPGAs for flexibility and to keep the cost of the demonstrator acceptable.

The system’s memory model gives rise to additional requirements. Shared-memory, cache-coherent HPC multiprocessor systems, whether uniform or non-uniform memory access, rely on hardware support to enforce memory coherency and consistency at cache-line granularity. The consequences for the ICTN are: (i) Fine-grained communications. Typical coherent transactions comprise a total payload of 100–300 B, excluding the ICTN-internal overhead. (ii) Ordering of transactions must usually be strictly enforced. Although costly to implement in hardware, the sequential consistency model is generally accepted, so supporting it in the ICTN is a strong benefit. (iii) Efficient support for multicast and broadcast is a basic requirement for snooping symmetric multiprocessors as well as collective and synchronization operations—barriers, reductions, locks—used in HPC applications.

II. FROM REQUIREMENTS TO ARCHITECTURE

In this section, we provide a top-down exposition of the design decisions that shape the OSMOSIS architecture (Fig. 1), starting from the set of core requirements.

This research is supported in part by the University of California under subcontract number B527064.

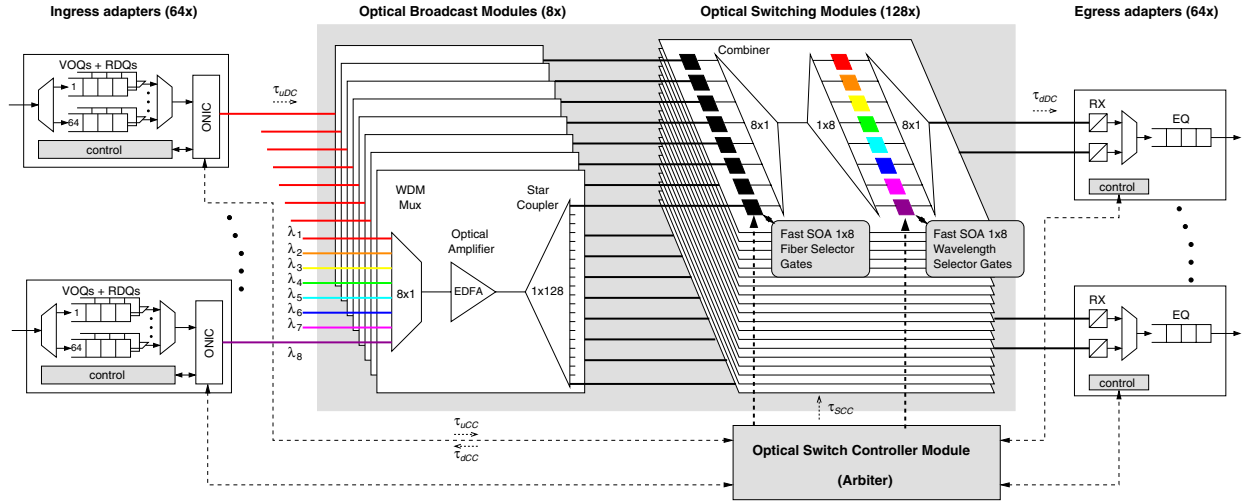


Fig. 1. OSMOSIS architecture.

A. Switch Degree

The most important requirement in an HPC ICTN is low latency, which is typically measured application–application across the network. As OSMOSIS must support 2048 nodes, we must design the system to achieve low end–end latency even with that many nodes. As scaling a single-stage network to 2048 nodes is generally not cost-effective because of the quadratic complexity involved, we must turn to multi-stage topologies. Here, we have a choice between *direct* (k -ary n -cubes: mesh, torus, hypercube) and *indirect* (multi-stage interconnection networks (MIN): Benes, k -ary n -fly, fat tree) topologies. Although direct networks scale very well to large node counts using small degree switches, the number of hops and therefore the worst-case latency between remote nodes grow quickly. Given our challenging $1\text{-}\mu\text{s}$ latency target, we therefore opt for an MIN topology, namely a fat tree, which has the advantages of being non-blocking, offering high path diversity and short paths to neighbouring nodes.

A fat-tree network with S levels can scale to $M = N(N/2)^{S-1}$ nodes using $(N(S-1) + N/2)(N/2)^{S-2}$ switches, where N is the (even) switch degree. Inverting this expression, we can express the switch degree N as a function of the number of nodes M and the number of levels S as follows: $N = \lceil 2 \sqrt[S]{M/2} \rceil$. Table I shows the required switch degree for various values of M and S . The maximum number of hops in an S -level fat tree equals $2S - 1$.

We have selected $N = 64$, which allows scaling to 2048 nodes using 96 switches in just two levels, i.e., there are at most three hops between any two nodes. Moreover, with one

TABLE I
N AS A FUNCTION OF M AND S

M	S						
	1	2	3	4	5	6	7
512	512	32	14	8	8	6	6
2048	2048	64	22	12	8	8	6
8192	8192	128	32	16	12	8	8
64 K	64 K	364	64	28	16	12	10

additional stage we can scale to 64 K nodes.

B. Switching paradigm

The latency and efficiency requirements are the main determining factors in our choice of *cell switching*. In this respect OSMOSIS is fundamentally different from most other optical switches, which are mostly based on techniques such as meshing, circuit switching, provisioning (bandwidth reservation), or aggregation (burst or container switching). These approaches either are prohibitively expensive (meshing), too inefficient (meshing and circuit switching), or introduce too much latency (aggregation). Instead, OSMOSIS switches fixed-size cells by reconfiguring the optical core on a cell-by-cell basis. The key enabler is the fast switching time (few ns) achieved by state-of-the-art *semiconductor optical amplifiers* (SOAs), which now, for the first time, allows low-overhead optical cell switching.

C. Cell format

Table II shows the fixed-size OSMOSIS cell format. Its overhead comprises the SOA switching time, preamble, cell header, forward error correction (FEC), line coding, and jitter budget. To minimize switch latency, the demonstrator data and control channels are run synchronously. We use standard Sonet/SDH components for the 40 Gb/s data channel and InfiniBand PHY components for the 2.5 Gb/s control channel. The different frequency requirements of these components determine the smallest possible cell size of 2048 bits, i.e., a time slot of 51.2 ns. In general, shorter cells are more desirable: they offer lower latency, finer granularity, and less padding overhead. Close to our 75% objective, the 256 B cell results in an efficiency of 73.4%. The OSMOSIS demonstrator can sustain full link bandwidth with traffic consisting only of 256 B cells. Moreover, this cell size is well suited to cache-coherent shared-memory systems (Sec. I-B).

D. Queuing and arbitration

SOAs lend themselves well to implement the optical equivalent of a *crossbar* using a *broadcast-and-select* (B&S) datapath architecture combining space- and wavelength-division

TABLE II
OSMOSIS CELL FORMAT DETAILS

field	purpose	(ns)	(bits)
Content with physical layer dependencies (estimates)			
D	SOA switchover time	4	160
S	Receiver TIA and PLL phase sync time	2.5	100
A	Cell boundary jitter and alignment mark	1.5	60
P	Physical channel coding for F, H, and U	0.8	32
Content without physical layer dependencies			
F	FEC for F,H,U, and P	3.2	128
H	Cell data header (multi-stage compliant)	1.6	64
U	Cell user data (73.4% of 2048 bits)	37.6	1504
total		51.2	2048

multiplexing (Sec. III and [1]). The advantages of a B&S structure are that it is fully non-blocking, inherently offers broadcast capability, and can (independently) scale in the space- and the wavelength dimensions. The crossbar equivalence heavily influences the queuing and arbitration aspects.

1) *Buffer placement*: As optical buffering is not yet mature, we cannot resolve contention for such a large-degree switch in the optical domain, so we need electronic buffers. We place buffers at the inputs as well as the outputs of the optical core, which results in a combined-input-output-queued (CIOQ) architecture. To eliminate head-of-line (HOL) blocking at the input buffers, we arrange them in a virtual-output-queuing (VOQ) fashion. These buffers are located in the *adapter card* of the corresponding port, which performs all interfacing functions between the attached computing node and the ICTN. We distinguish between the *ingress* and the *egress* adapter, although they are typically co-located on the same card. The adapter card provides an interface to the host channel adapter (HCA) of the computing node.

2) *Maximum throughput*: The maximum throughput is a relative measure that expresses how much of the aggregate raw data rate can be used without saturating (overloading) the switch. This is determined by the traffic pattern, which we cannot influence, and the arbitration-related aspects of the control path, i.e., the queuing structure, the control-channel protocol, and the arbitration algorithm. The first step towards high throughput is making sure that HOL blocking cannot occur, which is accomplished by the VOQ organization of the input queues. The second step is making sure that the arbiter has an accurate view of the current VOQ state, which we accomplish by means of a special protocol (Sec. II-E.2). The third and final step is using centralized, efficient arbitration.

3) *Arbitration*: The cell-switching nature of the switch implies that, to achieve high maximum throughput and low latency, it needs a centralized arbiter to compute a high-quality, i.e., near-optimal, matching between the inputs and the outputs. This problem has been studied extensively, and there are a number of well-known practical, heuristic, iterative algorithms [2], [3] to solve it. This class of algorithms achieves up to 100% throughput under uniform traffic patterns, although not strictly work-conserving for any arbitrary traffic pattern. However, they need about $\log_2(N)$ iterations to converge on a maximal matching. The challenge is to perform the full arbitration process once in every time slot of 51.2 ns. This problem is compounded by the large switch degree, which

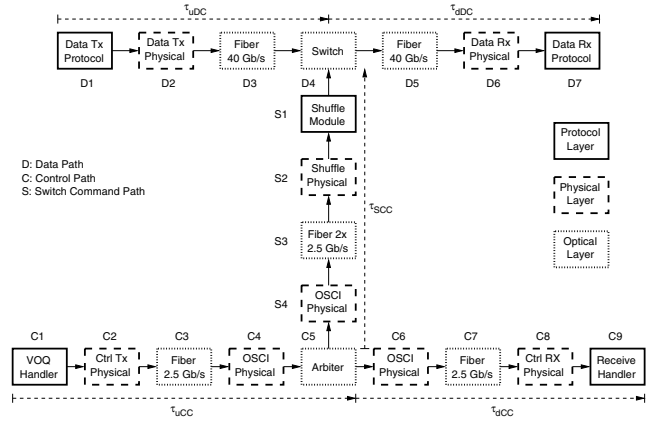


Fig. 2. Latency components

implies that not only does every iteration take longer (because more ports must be considered) but also that more iterations are required to achieve satisfactory latency-throughput performance characteristics. To increase both the arbitration rate and quality, we employ our previously published [4] *fast low-latency parallel pipelined arbitration* (FLPPR) scheme, which operates a pipeline of identical sub-arbiters.

E. Distribution

1) *Latencies*: In high-capacity, high-degree switch implementations, physical distribution plays an important role [5]. The resulting round-trip (RT) times in OSMOSIS are significant enough to impact key aspects of the design. In particular, we must cope with latencies on the up- and downstream data channels (τ_{uDC} , τ_{dDC}), on the up- and downstream control channels (τ_{uCC} , τ_{dCC}), and on the switch command channel (τ_{SCC}). Figure 2 shows the major latency components. For the OSMOSIS FPGA-based demonstrator, we estimate these latencies as follows (excluding any fiber): $\tau_{uDC} = 386$ ns, $\tau_{dDC} = 278$ ns, $\tau_{uCC} = 376$ ns, $\tau_{dCC} = 184$ ns, $\tau_{SCC} = 418$ ns. Hence, the total latency for one cell transmission across the switch equals 1224 ns, which corresponds to the sum of the full control path ($\tau_{uCC} + \tau_{dCC}$) and the full data path ($\tau_{uDC} + \tau_{dDC}$). In comparison, we estimate the latency of the same optical switch system when implemented with ASIC components at 280 ns. Note that the latency of a two-stage fat tree corresponds to three times the above single-stage latencies, and that 4.93 ns must be added per meter of fiber.

2) ΔRGP : We have shown in [6] that a long RT has a negative performance impact and that an *incremental request-grant protocol* (ΔRGP) can mitigate this impact. ΔRGP communicates relative VOQ state updates (encoded with $\log_2(N)$ bits) rather than an absolute request vector (N bits). Consequently, the arbiter must keep track of the number of pending requests per VOQ. Upon cell arrival, the input issues a request for the corresponding output to the arbiter. In every time slot, the arbiter processes the incoming requests by incrementing the corresponding VOQ counters, computes a matching, returns the corresponding grants, and decrements the corresponding VOQ counters. When an adapter receives a grant, it dequeues

the HOL cell of the VOQ granted and forwards it to the crossbar. The control messages carry the fields needed for Δ RGP. The arbiter-side functions are part of the *control channel interface* (CCI) units on the arbiter board (Sec. IV).

3) *Synchronicity*: The size and latencies of the system also make it difficult to ensure that the entire system is cell-synchronous. We employ a central reference clock that we distribute to all modules via the InfiniBand physical layer of the control channel (Sec. IV-D).

F. Reliability

To meet the objective of an error rate better than 10^{-21} , we have adopted a two-tier approach to reliable delivery (RD). As the raw bit-error rate of the optical path is expected to be near 10^{-10} , we will employ an FEC on the header and data part of the packet to reduce the bit-error rate to an estimated 10^{-17} . Any errors detected but not corrected by the FEC are handled by the RD scheme. We employ a window-based link-level retransmission scheme that performs up to three retries per packet. This will reduce the probability of a packet not being delivered correctly at the application level to much less than 10^{-21} . Together, these schemes meet the extremely low error-rate requirement.

We distinguish between intra-switch RD (Sec. IV-F) and inter-switch RD (Sec. III-B). Intra-switch RD covers all data paths from any ingress adapter to any egress adapter within one OSMOSIS switching node (N^2 connections in total), whereas inter-switch RD covers the links from the egress adapters of one stage to the ingress adapters of the next stage (N connections in total).

1) *Data channel coding*: We solve the challenge of coding on the data channel in three stages. The system requirements force us to find an optimal balance between delay and processing speed, and efficiency and complexity. The three stages are: The *physical line code* (PhLC), which is needed to satisfy the clock and data recovery (CDR) units at the receiving side. It provides a defined maximum run-length and a maximum DC offset in the data bit stream. The run-length limitation guarantees a certain rate of transitions for the CDR, whereas the DC offset has to be limited to keep the receiver diode and its transimpedance amplifier (TIA) in its working window. The *alignment marker* (AM) is a unique pattern in the data stream to lock on at the head of a cell. As the CDR loses its lock at the end of every cell, the word alignment of the next cell's data is found by this pattern, which is crucial for the FEC coding. The third stage is an FEC capable of correcting single-bit and detecting double-bit errors in a single cell. Many higher-order errors can be detected or even corrected. This three-stage approach creates about 4% overhead for the PhLC, 1.6% for the AM, and 6% for the FEC. The AM can be processed on the fly, whereas the PhLC and FEC lead to latencies of about one and four cell cycles, resp. These numbers are for an FPGA-based implementation. A later integration in gate-array technology would reduce the latency estimates as well as the overhead by a factor of four.

2) *Control path*: We must ensure that the Δ RGP protocol remains consistent in the presence of errors, i.e., the actual

VOQ state at the line cards should be consistent with the state maintained by the arbiter within the bounds given by the RT. To this end, we have introduced a *census mechanism* [7] that determines whether the state of a specific VOQ is consistent, and, if not, indicates the magnitude of the error. This census mechanism is also incorporated in the control message format.

III. OSMOSIS ARCHITECTURE

Now that we have explained the rationale behind the most important design choices, we assemble the pieces and present the overall architecture.

A. Single-stage architecture

Figure 1 shows the high-level OSMOSIS switch architecture in the single-stage configuration. It supports 64 nodes and operates in a synchronous, time-slotted fashion with fixed-size cells. Computing nodes are attached to the interconnect via adapter cards, which perform the interfacing functions between the computing nodes and the ICTN and include an *optical network interface card* (ONIC), which performs the electro-optical (E/O) and opto-electrical (O/E) conversions. Every adapter comprises a full set of N VOQs with a corresponding set of N reliable delivery queues (RDQs), as well as an egress queue (EQ). Every adapter has a dedicated optical control link to the centralized arbiter, which carries the control channel protocols. The arbiter is located on a separate card close to the optical switching modules (OSMs); it configures the OSMs via the *switch command channels* (SCCs).

The routing function is implemented with a B&S architecture that combines eight-way space- and eight-way wavelength-division multiplexing to implement a 64-port fabric. Every adapter is assigned a specific wavelength λ_i ; the adapters are grouped by eight such that every adapter in a group has a unique wavelength. The optical signals per group are multiplexed onto a single fiber. First, the resulting signal is amplified by an erbium-doped fiber amplifier (EDFA), before being broadcast by a 128-way coupler to 128 OSMs. Every OSM has a fiber- and a color-selection stage. The fiber-selection stage consists of eight SOAs to select the fiber that carries the cell to be received. A planar lightwave circuit (PLC) combines the output of all SOAs to a common port, and the resulting stream is then demultiplexed and passed through the color-selection stage (also consisting of eight SOAs), which selects the desired color. The stream is then re-multiplexed onto a common output port and delivered to the broadband receiver residing at the egress ONIC. Synchronously with cell transmission, the centralized arbiter performs contention resolution on requests delivered to it via optical control channels, and provides the control signals (via the SCC) that drive the fiber- and color-selection SOAs just as the cells arrive at each OSM. Further details of the architecture and technology of OSMOSIS can be found in [1].

A special feature of OSMOSIS is the presence of two OSMs and two receivers per egress adapter, which allows up to two cells to be delivered to the same adapter in one time slot, i.e., the output speed-up equals two. Hence, the crossbar is asymmetric: it has 64 inputs and 128 outputs. The required

rate conversion is handled by the egress buffers. The objective of this arrangement is to improve performance; the exact mode of operation will be the subject of a follow-on paper.

To prevent overflow of the egress buffers, we employ an on-off flow control (FC) loop between the egress buffers and the arbiter, which is embedded in the upstream control channel messages. Whenever the permission for a specific egress buffer is removed, the arbiter will no longer issue any grant for the corresponding output until the permission is reinstated.

B. Multi-stage support

As argued in Sec. II-A, we employ a fat tree network to scale to 2048 or more nodes. Here, we discuss the provisions in OSMOSIS required to support this. First, the cell header comprises source-routing tags for up to three stages, link-level on-off FC (LL-FC), as well as sequence numbers of acknowledgments (ACKs) for link-level inter-switch RD.

The cell-switching nature of OSMOSIS implies that contention will occur at every stage, hence buffering is required at every stage. Therefore, despite the cost, power and latency penalty this entails, it is inevitable that there be electronic buffers between the stages, necessitating O/E/O conversions. The network must also be bidirectional to enable in-band, closed-loop LL-FC for all connections between subsequent stages to achieve lossless communication.

At every stage, input and output ports exist in pairs implemented in the same physical adapter card. The LL-FC for the upstream direction is embedded in the headers of the cells traveling downstream and vice versa. The receiving ingress adapter extracts the LL-FC and passes it to its paired egress adapter; the ingress adapter also informs the egress adapter of the FC status depending on its current buffer state, so that the egress adapter can include this information in the next outbound cell header (which may be an idle if there is no data traffic). The FC information is derived from a programmable threshold, which should be set low enough to avoid buffer overflow and high enough to prevent underflow, i.e., at least one RT below full and at least one RT above empty.

IV. CONTROL PATH IMPLEMENTATION

In this section, we highlight the challenges and technical trade-offs made for the control section of the OSMOSIS system. We explain how we map the solutions proposed in Sec. II to a distributed physical implementation, and demonstrate its feasibility using a set of 45 FPGAs. Fig. 3 illustrates this implementation, which we refer to as the *optical switch controller module* (OSCM). We explain the components shown, namely, the *control channel interfaces* (CCIs), the *switch command interfaces* (SCIs), the arbitration (FLPPR), and the clocking and control unit. We mainly focus on the arbitration—the most challenging part.

A. Centralized crossbar arbiter

Round-robin (RR) based matching algorithms are widely used in industrial products because they are amenable to fast hardware implementations. We started our feasibility studies by evaluating an iterative RR-based matching algorithm for

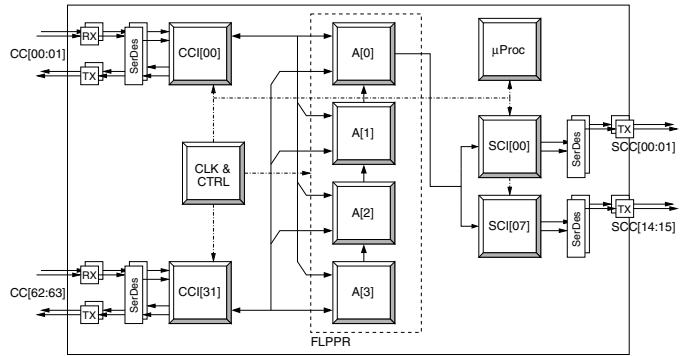


Fig. 3. Logical OSCM architecture. The shaded boxes represent practical chip boundaries.

TABLE III
SIZING IN XILINX VIRTEX-II-PRO (SPEED GRADE -6).

N	device utilization		performance		#iterations		
	slices	%	#nets	f_{\max} (MHz)	t_{\min} (ns)	I_a	I_t
iSLIP (in- and output selectors)							
4	266	0.60	2,075	203.9	4.90	10.4	2
8	1,071	2.43	8,008	119.1	8.39	6.1	3
16	4,544	10.3	33,770	79.9	12.51	4.1	4
32	15,046	34.1	114,987	57.9	17.27	3.0	5
48	34,652	78.6	264,174	42.4	23.58	2.2	5.6
52	41,437	93.8	316,856	44.1	22.66	2.3	5.7
64	does not fit FPGA device		—	—	—	—	6
DRRM (output selectors)							
64	30,610	69	249,267	40	25	2	6

the implementation of our crossbar arbiter. Specifically, we selected the well-known iSLIP [2] algorithm, which was designed to achieve an improvement in terms of efficiency, latency, fairness and speed compared with existing algorithms.

1) *Sizing assumptions*: Our target technology is the biggest and fastest FPGA available from Xilinx at the time of implementation, namely, the “xc2vp100-6ff1704,” a Virtex-II-Pro series FPGA providing 8 M system gates (100 K logic cells)¹ and 1040 users I/Os. We used release 8.0 of the Synplify Pro synthesis software from Synplicity, and release 6.3 of the Integrated Software Environment software suite from Xilinx.

2) *Physical implementation of a three-phase matching algorithm*: Table III presents the sizing measurements for the unconstrained placement and routing of the request-grant-accept phases of iSLIP, based on the pipelined implementation described in [8]. These sizing numbers only represent the core of the algorithm without the I/O interfaces required to convey the external requests and grants to/from the arbiter device. The table lists the device utilization in the number of slices, percentage and number of nets, the arbiter performance in terms of maximum clock frequency f_{\max} and minimum clock period t_{\min} , and the number of achievable iterations I_a vs. the target $I_t = \log_2(N)$.

The device utilization results of Table III show that the largest arbiter feasible in a single Virtex-II-Pro xc2vp100 is somewhere in the range of 52×52 . One possible solution to

¹Virtex logic cell = (1) 4-input LUT + (1) flip-flop + carry logic. Virtex slice = 2 logic cells.

overcome the limitations of the monolithic implementation would be to distribute the N input (accept) and N output (grant) selectors across two separate devices. However, the bisectional bandwidth ($O(N^2)$) across the chip boundaries renders this approach unpractical for large N . Therefore, we cannot use a monolithic three-phase matching algorithm such as iSLIP, FIRM, or SRRM for our centralized crossbar arbiter, because it exceeds the capacity of the largest FPGA device currently available.

3) *Physical implementation of a two-phase matching algorithm:* The dual RR matching (DRRM) scheme [3] achieves a performance identical to that of iSLIP but is easier to distribute. Contrary to iSLIP, DRRM is a two-phase algorithm. In phase one, instead of broadcasting all requests to the output selectors, DRRM employs a selector per input that selects one request and forwards it to the corresponding output selector. In phase two, every output selector grants one of the requests (if any); there is no need for a third (accept) phase. Although DRRM has the same number of selectors as a three-phase algorithm ($2N$), it has a much lower bisectional bandwidth ($O(N \log_2(N))$) owing to its simpler request-grant protocol. In our case the bisectional bandwidth amounts to an acceptable 7.5 Gb/s ($N = 64$, $T_c = 51.2$ ns), and therefore enables the physical separation of the input and output selectors into different FPGA devices.

Table III presents the sizing measurements for the constrained placement and routing of the output selectors for a two-phase arbiter with $N = 64$. In this case, the sizing numbers represent the finalized FPGA, i.e., including all I/O interfaces required to convey the external requests and grants to/from the arbiter device as well as all the status and configuration registers required to control the arbiter device. This sizing experiment shows that this is feasible, but that we can only expect to complete two iterations in one time slot, which is well short of the target of six.

B. FLPPR

As mentioned in Sec. II-D, we employ the FLPPR pipelined arbiter architecture to increase the number of iterations per matching. This scheme specifically eliminates the latency penalty associated with existing pipelining methods. We opt for four identical sub-arbiters (shown in Fig. 3 as $A[0] \cdots A[3]$), each producing two iterations per time slot. The operation and performance of FLPPR have been shown in [4].

C. Control channel interface (CCI)

As the four sub-arbiters of the FLPPR arrangement only implement the output selectors of DRRM, a similar amount of FPGA resources are required to implement the input selectors. Instead of implementing these in a set of four additional centralized FPGAs, we opted for a more distributed approach in which the input selectors share physical devices with the control channel (CC) logic on the arbiter side. This logic includes the link management unit, Δ RGP, the census mechanism [7], the RD protocol, and the egress buffer FC. We selected a high-end Altera Stratix EP1S80 FPGA for the CCI. Our area and timing investigations for the shared

implementation of these six functions indicate a total device area utilization of 20% per CCI. Because of a limitation in the number of enhanced phase-locked loops (PLLs) available for the CCI FPGA, we have to limit the integration to a maximum of two CCs per CCI, for a total of 32 CCIs to handle all 64 CCs. Because of PCB area limitations, we implement every CCI on an individual (22×15 cm²) daughterboard, referred to as *optical switch controller interface* (OSCI) card.

D. System clocking and synchronization

An unbuffered crossbar switch must operate in a cell-synchronous fashion. This requires a system-wide distribution of a phase-synchronized reference frequency (156.25 Mhz, 8 clock cycles per cell) and a cell start signal (51.2 ns). We use a “time zero” beacon as a global real-time reference, and some defined control characters for the purpose of initializing and signaling. All synchronization is distributed from a central high-precision clock on the OSCM. The short switching time between cells requires a high-quality synchronous clock over the entire system. The implementation is in differential PECL (positive emitter-coupled logic) at the module level, and optical between the modules. Each component in the system is equipped with a timer macro, which consists of a synthesizer PLL, a real-time counter register, a configuration unit, and an RT delay analysis unit. The configuration unit presets the synthesizer PLL and the real-time counter based on programmable settings or the results of the RT analysis. This procedure maintains the timing accuracy over the entire system. Future developments in PLCs will lead to the replacement of some electronic components, which will greatly simplify the system at even higher accuracy.

E. Switch command interface (SCI)

The switch command interface (SCI) implements the physical layer interface from the arbiter (OSCM) to the OSMs. Every OSM can receive one switch command of size $\log_2(N) + 1$ in every time slot of 51.2 ns, for an aggregate throughput of $\frac{128 \cdot 7}{51.2} = 17.5$ Gb/s. To optimize design time and cost, we re-use the OSCI card and its programmable FPGA, which we refer to as SCI when it is configured to interface with the SCC. Each OSCI implements two physical channels, each providing 2 Gb/s of user data rate. We use eight OSCIs to transport all 128 commands to the OSMs. The SCI FPGA also implements a programmable delay line that is used to precisely match the latencies of the data, control and switch command channels to ensure correct and timely switching.

F. Intra-switch reliable delivery

It is the responsibility of the RD mechanism to ensure that the data flow from any input adapter to any output adapter in the switch is in-order and free of errors. For error detection, we rely on the FEC (Sec. II-F.1). Cells that the FEC cannot correct must be retransmitted; to this end, the sender stores cells in a *reliable delivery queue* (RDQ) until they are acknowledged by the receiver. As we have already described our approach to intra-switch RD in detail in [1], we only recall the key points.

Instead of piggybacking cumulative ACKs on the optical data channel, we have opted to use the control channel

path for relaying ACKs. This has several advantages. First, the ACK overhead is moved from the data channel to the control channel, which helps in meeting the 75% efficiency requirement. Second, it provides a fixed-bandwidth, fixed-latency path for the ACKs, whereas piggybacking on the data channel relies on the presence of corresponding data traffic in the reverse direction. In the absence of such traffic, the sender may have to buffer a large number of unacknowledged cells. Instead, using the control channel allows ACKs to reach the sender in every time slot, independent of the data traffic, with *deterministic* latency. Third, this allows us to acknowledge every data cell individually.

We employ a Go-Back-N (GBN) retransmission policy, which is consistent with burst errors seen in optical links. We do not use resequencing because the error rate after FEC is so low that the bandwidth wastage due to an occasional retransmission of the full GBN window does not justify the added hardware complexity.

This approach requires control messages to relay ACKs from egress to ingress through the arbiter. However, our requirements for high data-channel efficiency and throughput justify the additional hardware, wiring, and control channel bandwidth required to relay ACKs. Synthesis of the VOQ, RDQ, and census logic for an Altera Stratix EP1S80 device results in a sizing of 31,616 Altera logic elements (40% utilization) and $f_{\max} = 71.4$ MHz ($t_{\min} = 14$ ns).

G. OSM packaging

The sizings in the preceding sections show that the OSMC comprises 32 CCI chips, eight SCI chips, four arbiter chips, one ACK routing chip, and one clocking and control chip, for a total of 45 FPGAs. In addition, there is an embedded microprocessor for configuration purposes. The CCI and SCI FPGAs reside on the OSCI cards. The remaining FPGAs reside on a large (53×43 cm²) midplane, referred to as *optical switch controller board* (OSCB), which in addition has 40 slots for the OSCI cards. Figure 4 shows the assembly of prototype OSCB and OSCI.

V. CONCLUSIONS AND FUTURE WORK

The OSMOSIS project explores the use of optical switching in HPC interconnects. We showed how the core requirements—in particular low latency in combination with high scalability—lead to a cell-switching, high-degree, virtual-output-queued, centrally-arbitrated switch; its all-optical data path features a broadcast-and-select architecture that combines space- and wavelength-division multiplexing, and employs fast SOAs as the key enabling optical technology.

We identified the major challenges faced in the implementation of the OSMOSIS control path, taking into account that we can only use FPGAs. We introduced a highly distributed arbiter implementation that solves the technical issues. Moreover, we showed that this implementation is feasible and can meet the target requirements.

In future work, we will address support for multicast and virtual lanes, describe the distributed arbiter implementation in more detail, and consider commercialization aspects. A

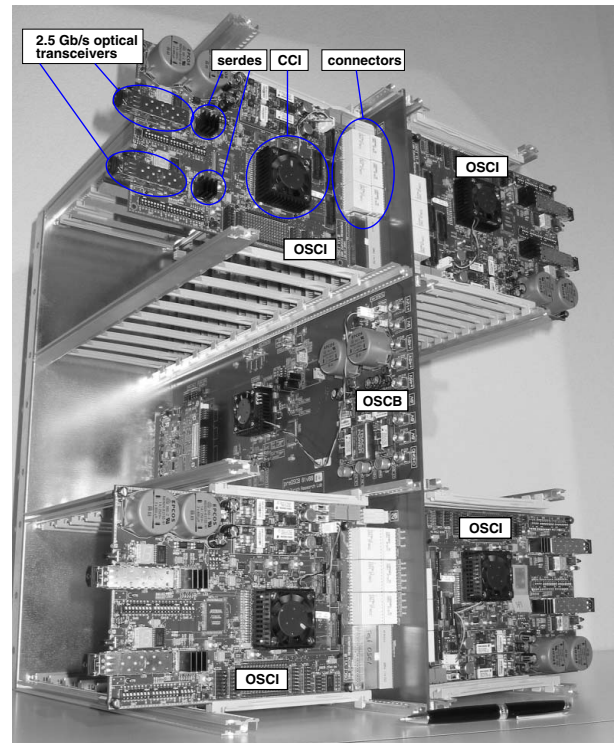


Fig. 4. Physical arbiter (OSCM) demonstrator packaging

preliminary scaling study of the OSMOSIS control path shows that the system can gain at least $4 \times$ in an ASIC implementation, enabling either faster ports (160 vs. 40 Gb/s), smaller cells (64 vs. 256 B), or a combination thereof.

ACKNOWLEDGMENTS

The authors thank the sponsors and acknowledge the technical contributions of everybody involved at IBM, Corning, Photonic Controls, and G&O.

REFERENCES

- [1] R. Hemenway, R. Grzybowski, C. Minkenberg, and R. Luijten, "Optical-packet-switched interconnect for supercomputer applications," *OSA J. Opt. Netw.*, vol. 3, no. 12, pp. 900–913, Dec. 2004.
- [2] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Networking*, vol. 7, no. 2, pp. 188–201, Apr. 1999.
- [3] H. Chao and J. Park, "Centralized contention resolution schemes for a large-capacity optical ATM switch," in *Proc. IEEE ATM Workshop*, Fairfax, VA, May 1998, pp. 11–16.
- [4] C. Minkenberg, I. Iliadis, and F. Abel, "Low-latency pipelined crossbar arbitration," in *Proc. IEEE GLOBECOM 2004*, Dallas, TX, Dec. 2004.
- [5] C. Minkenberg, R. Luijten, F. Abel, W. Denzel, and M. Gusat, "Current issues in packet switch design," *ACM Computer Commun. Rev.*, vol. 33, no. 1, pp. 119–124, Jan. 2003.
- [6] C. Minkenberg, "Performance of i-SLIP scheduling with large round-trip latency," in *Proc. IEEE Workshop on High-Performance Switching and Routing HPSR 2003*, Torino, Italy, June 24–27, 2003, pp. 49–54.
- [7] C. Minkenberg, F. Abel, and M. Gusat, "Reliable control protocol for crossbar arbitration," *IEEE Commun. Lett.*, vol. 9, no. 2, pp. 178–180, Feb. 2005.
- [8] P. Gupta and N. McKeown, "Designing and implementing a fast crossbar scheduler," *IEEE Micro*, vol. 19, no. 1, pp. 20–28, Jan./Feb. 1999.