Parallel Modular Scheduler Design for Clos Switches in Optical Data Centre Networks

Paris Andreades, *Student Member, IEEE,* and Georgios Zervas, *Member, IEEE*

***Abstract*—As data centers enter the exascale computing era their internal traffic, exchanged between network nodes, increases exponentially. Optical networking is an attractive solution to deliver the high capacity, low latency and scalable interconnection needed. Amongst other switching methods, packet switching is particularly interesting as it can be widely deployed in the network to handle rapidly-changing traffic of arbitrary size. Nanosecond-reconfigurable photonic integrated switch fabrics, built as multi-stage architectures such as the Clos network, are key enablers to scalable packet switching. However, the accompanying control plane needs to also operate on packet timescales. Designing a central scheduler, to control an optical packet switch in nanoseconds, presents a challenge especially as the switch size increases. To this end, we present a highly-parallel, modular scheduler design for Clos switches along with a proposed routing scheme to enable nanosecond scalable scheduling. We implement our scheduler as an application-specific integrated circuit (ASIC) and demonstrate scaling to a 256 x 256 size with an ultra-low scheduling delay of only 6.0 ns. In a cycle-accurate rack-scale network emulation, for this switch size, we show a minimum end-to-end latency of 32.0 ns and maintain nanosecond average latency up to 80% of input traffic load. We achieve zero packet loss and short-tailed packet latency distributions for all traffic loads and switch sizes.**

***Index Terms*—IEEE, IEEEtran, journal, LaTeX, paper, template.**

I. Introduction

THE global traffic within data centres is estimated to reach 14.7 zettabytes per year by 2021, driven by technology trends such as cloud computing and data centre virtualization [1]. Big data processing, workload migration, storage replication and retrieving data residing on multiple hosts are all examples of tasks that require data exchange between data centre machines [1]–[3]. The growth of this so called East-West traffic places stringent requirements on the network latency and bandwidth, causing a shift from the traditional hierarchical tree to the flatter leaf-spine network topology [4], [5], shown in Fig. 1. Optical fibre is typically installed between the leaf and spine layers to establish low loss and high bandwidth point-to-point connections. Nonetheless, the switches themselves are still electronic, limiting further performance scaling. Commercial electronic switches have a delay on the order of 200 ns [6] and limited capacity due to the number of high-speed pins available on the switch chip or the number of connectors fitting on a rack unit front panel [7]. These limitations can be addressed by optical switching reconfigurable in nanoseconds with orders of magnitude higher capacity, using wavelength-division multiplexing (WDM).

Optical switches built as micro-electro-mechanical systems (MEMS) have been widely proposed for data centre networks. Notable examples include the RotorNet [8], c-Through [9], Helios [10], Proteus [11] and Mordia [12] prototypes. However, because they are reconfigurable in milliseconds, they are used for circuit switching at the higher network layers where there is a larger traffic volume or in networks with slowly changing traffic. Hence, they are intended to be used alongside an electronic packet-switched network which would handle any bursty and rapidly-changing traffic. Moreover, their control plane needs to perform traffic demand estimation prior to circuit allocation to increase utilization [12], which could incur hundreds of milliseconds additional delay.

Optical packet switching has also been proposed based on nanosecond-reconfigurable photonic integrated architectures built using micro-ring resonators [13], [14], semiconductor optical amplifiers (SOAs) [15], [16], Mach-Zehnder interferometers (MZIs) [17] or a combination of these technologies [18]–[20]. Such switches could be deployed at any network layer without any limitation on the traffic size and stability. However, designing a switch control plane that operates on packet timescales, as the switch size scales, still remains a challenge in optical packet switch design. The Data Vortex [21], SPINet [22], OSMOSIS [23], the OPSquare switch as well as the work in [24], [25], all target scalable nanosecond switch control. In all cases, fast output-port arbitration for switch scheduling is key in implementing low-latency control.

In previous work [26], we experimentally demonstrated a nanosecond control plane on field-programmable gate array (FPGA) boards. It enabled optical packet switching with a minimum 75 ns end-to-end latency in a 32-port SOA-crossbar system, for top-of-rack (ToR) application. In [27] a new output-port arbitration circuit for the switch scheduler was presented, doubling the switch size for the same scheduling delay. In [28], [29] we designed schedulers for optical switches built in a Clos network topology [30], as shown in Fig. 1, which is practical for photonic integration [31]. Also, this modular switch architecture enables scaling to large sizes while maintaining a low scheduling delay.

This work expands on [29] by implementing the central Clos switch scheduler on hardware. Moreover, the scheduler was redesigned as a parallel and modular hardware structure to allow for clock speed optimization. The scheduler module designs

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R035342/1 and in part by the EU Horizon 2020 programme (Industrial Leadership section) under Grant 687632.

Paris Andreades and Georgios Zervas are with the Electronic and Electrical Engineering Department, University College London (UCL), London, WC1E 7JE, UK (e-mail: paris.andreades.09@ucl.ac.uk, g.zervas@ucl.ac.uk).

Manuscript received Month dd, yyyy; revised Month dd, yyyy.

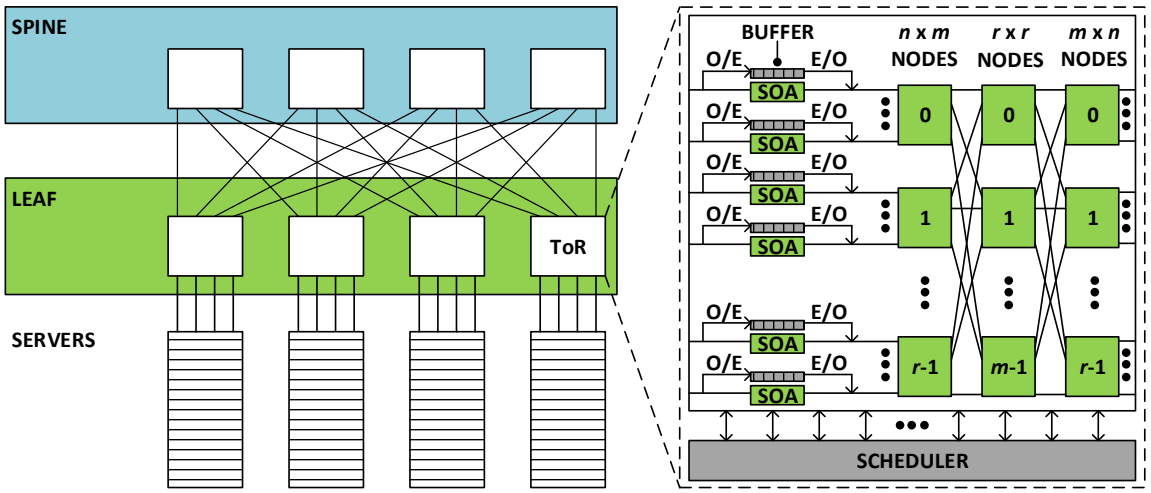

Fig. 1. A leaf-spine data centre with the proposed optical top-of-rack (ToR) Clos switch. Conversion from optical-to-electrical (O/E) and electrical-to-optical (E/O) enables electronic buffering at the switch inputs, when switch path contention occurs. The electronic scheduler processes switch path requests from the network interfaces at the servers and at the switch buffers to reconfigure the optical switch in nanoseconds.

are presented and discussed in detail and are individually synthesized as application-specific integrated circuits (ASICs) in a 45 nm CMOS process. The results verify scheduling a 256×256 switch in 6.0 ns and identify the critical path module, which determines the scheduler minimum clock period. That module is then implemented on a Xilinx Virtex-7 XC7V690T FPGA board to compare our work with some of the fastest switch designs in the literature. The switch latency and throughput performance are evaluated in a cycle-accurate emulation of our proposed system concept.

II. Related Work

In this section we look at different notable scheduling approaches to optical packet switch control from various research labs in the field. The switch architecture is important as it could simplify scheduling and reduce its impact on control delay but at the same time it should not increase the data plane complexity, hindering switch implementation. In general, scheduling can be executed centrally or in a distributed fashion, with the former usually considered to be of high complexity and incurring a considerable control plane delay. In this work, we aim to show otherwise. Table I compares the scheduling delay in different optical packet switches reported in the literature.

The Data Vortex [21], [32] and SPINet [22], [33] designs distribute scheduling to the 2×2 modules of an $N \times N$ banyan network, which scales by cascading $log_2 N$ stages. Scheduling per module is executed in 10.0 ns but cascading them in many stages increases total control delay and limits the switch throughput at large sizes.

The OPSquare wavelength-routed switch [34]–[36] is also modular and scales by stacking modules in a 2-stage Spanke architecture and by increasing the wavelength channels. It avoids arbitration for the output modules by using wavelength conversion. The arbitration time for the input modules depends on the number of wavelengths routed, bringing the total scheduling delay to 25 ns, independent of switch size. However, wavelength conversion and the high optical components count increase the implementation complexity and cost.

All aforementioned designs use in-band request schemes which have scheduling overheads for optical filtering, O/E conversion and request detection, in addition to arbitration. The design by Proietti et al. [24] avoids this by using optical instead of electronic scheduling. The switch is based on an $N \times N$ input-buffered arrayed waveguide grating router (AWGR) and scaling is determined by the highest port-count supported by the AWGR technology. Scheduling is optical and distributed to the output ports, where reflective semiconductor optical amplification (RSOA) is used for nanosecond arbitration, independent of N. However, scheduling is dominated by laser tuning time (8 ns), switch round-trip time (5 ns) and grant detection time (4 ns) at the input port. Also, arbitration fairness is degraded, especially as N grows, compared to electronic round-robin schemes.

The central schedulers in the OSMOSIS prototype [23] and in the work by Cerutti et al. [25] use parallel iterative algorithms to improve throughput in an $N \times N$ crossbar switch at the cost of high scheduler complexity and long delays. The scheduler design in our previous work [27], for crossbar switches, trades off throughput for ultra-low delay; reconfiguring a 64×64 crossbar in 18.8 ns. Nevertheless, an $N \times N$ crossbar is not practical for implementation at large sizes, due to the N^2 switch elements required to built it.

All designs that issue an acknowledgement (ACK) from the switch back to the packet source, whether in an optical or electrical form, incur control overheads for ACK transport, detection and processing at the source, in addition to scheduling.

This work aims at a nanosecond-reconfigurable switch design that scales both in the data plane and control plane. It is based on a 3-stage Clos architecture and a novel routing scheme for simplified nanosecond scheduling and practical

TABLE I
WORK COMPARISON - SCHEDULING DELAY

Switch Design	Switch Size			Architecture	Method	Request	ACK
	16x16	64x64	256x256				
This Work (ASIC)	**3.3** ns	**4.2** ns	**6.0** ns	Clos	Central	Out-of-band	No
Data Vortex/SPINet [32], [33]	≈10.0 ns			Banyan	Distributed	In-band	Yes
Proietti et al. [24]	17.0 ns			AWGR	Distributed	In-band	Yes
This Work (FPGA)	**8.7** ns	**13.5** ns	**21.0** ns	Clos	Central	Out-of-band	No
Andreades et al. [27]	7.0 ns	18.8 ns	-	Crossbar	Central	Out-of-band	No
OPSquare Switch [34]	25.0 ns			Spanke	Distributed	In-band	Yes
OSMOSIS [23]	-	51.2 ns	-	Crossbar	Central	Out-of-band	Yes
Cerutti et al. [25]	1 μs	-	-	Crossbar	Central	Out-of-band	No

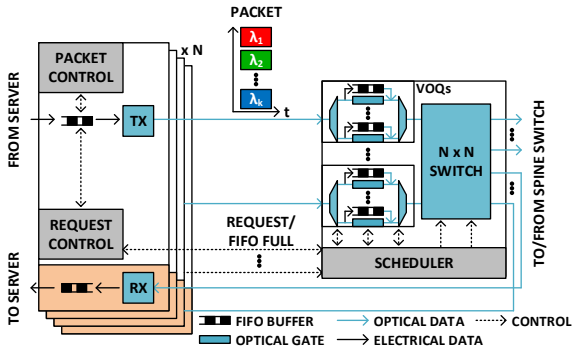

Fig. 2. System concept. Server-side "send and forget" network interfaces and optical top-of-rack (ToR) switch with input electronic buffers to avoid packet loss. Every transmitted packet is divided into *k* wavelengths to further reduce latency. The ToR scheduler reconfigures the switch in nanoseconds.

switch implementation. We can schedule a 256×256 Clos switch in just 6.0 ns in an ASIC implementation or in 21.0 ns in an FPGA implementation. The implementation results are presented in section V. Out-of-band electrical requests, processed by a central electronic scheduler, and speculative packet transmission eliminate other control overheads besides scheduling, as discussed in the next section.

III. SYSTEM CONCEPT

The proposed system concept is shown in Fig. 2. The $N \times N$ optical top-of-rack (ToR) switch interconnects N servers between themselves and to the spine switch. Internally, it is implemented as as Clos network. The server-side network interfaces implement a "send and forget" scheme whereby packets are transmitted speculatively without a guaranteed path through the switch, reducing in this way the control latency. Hence, the optical Clos switch implements electronic buffering at each input port to avoid packet loss for any failed path speculations, when at least two network interfaces contend for a switch path. The electronic scheduler processes path requests to allocate switch paths, resolves contention where necessary and reconfigures the optical switch to schedule traffic across it. In the next paragraphs the flow control is described in detail.

At every server network interface, the packets are initially buffered in a first-in, first-out (FIFO) policy. The request control reads the destination of the head-of-line (HOL) packet and issues a switch path request to the scheduler.

Once the request is sent, the packet control holds onto the HOL packet for a configurable number of hardware clock cycles before forwarding it to the transceiver for transmission. This allows for the scheduler to processes the request and reconfigure the switch, before the packet arrives there. Nanosecond scheduler delay, $t_{scheduler}$, is crucial for packet switching. This delay is fixed, dependent on the scheduler design, and the main focus of this work is to optimize the design for ultra-low delay (< 10 ns), as the switch size scales.

Unlike circuit-switched systems, where a packet is buffered at the source for possible retransmission until a path ACK (grant) is received, here every packet is forwarded to the transceiver speculatively, without a scheduler grant. Compared to the related work, this reduces the control delay by eliminating the overheads for grant transport from the switch back to the server network interface and then for synchronizing and processing it there. To avoid packet drop, in case of failed speculation, buffering is used at each switch input port.

At the transceiver, every packet is divided into *k* segments, each serialized at a fixed data rate onto a specific wavelength using, for example, a dedicated silicon photonic transceiver [37] integrated onto the network interface chip. Using wavelength-division multiplexing, the segments are transmitted as one unit in a wavelength-parallel packet structure to the switch input port, as shown in Fig. 2. This method, known as wavelength striping, increases the input port capacity (no. of wavelengths $\times$ data rate) and consequently reduces the packet (de)serialization delay (packet size $\div$ input port capacity). More importantly, by dynamically reconfiguring the number of wavelength channels bonded, based on the packet size, the system can support a low (de)serialization delay for variable-size packets.

The resulting wavelength-striped packet is transmitted on the WDM link to the switch input port, as shown in Fig. 2. In case of no switch path contention, hence no failed speculation, the wavelength-striped packet cuts through the already reconfigured switch and is received at the destination with the minimum end-to-end latency. Otherwise, the packet is stored at the switch input port.

At every switch input port, buffering is implemented as *N* parallel FIFO queues, one for each output port, also known

as virtual output queues (VOQs). Compared to using a single queue, the VOQ arrangement avoids the case where the HOL packet, while waiting to be switched to its destination, blocks the preceding packets in the queue even though they may be destined elsewhere. Thereby, throughput is improved and buffering delay is reduced, at the cost of increased scheduling algorithm complexity. Optical to electrical (O/E) and electrical to optical (E/O) conversions are required when a packet is stored or released from a VOQ.

The central scheduler therefore receives two sets of path requests; one from the network interfaces at the servers, in response to new packets, and one from the network interfaces at the switch input ports (not shown in the diagram), in response to switch VOQ packets. Then, it arbitrates access to switch paths, on *round-robin* basis, in parallel for the two sets and reconfigures the switch accordingly, including the VOQ control signals for storing and releasing packets. Priority is always given to VOQ requests for strict in-order packet delivery and low average latency.

As shown in Fig. 2, a "FIFO full" signal is asserted when a switch VOQ is occupied beyond a threshold value, to notify the corresponding server network interface to pause transmission (packets and requests). This provides control backpressure and enables managing a small VOQ size at the switch. For every switch input port there is an N-bit wide "FIFO full" bus, one bit for every VOQ, transmitted back to the server and overall this is the only control signaling from the switch to the servers. The VOQ threshold value depends on the propagation delay ($t_{propagation}$) between the switch and servers (round-trip) and also the scheduler delay ($t_{scheduler}$) during which the server packets are buffered. The threshold value is given by the following equation and determines the minimum VOQ size required to avoid packet loss up to 100% of traffic load:

$$\text{Threshold} = t_{scheduler} + 2t_{propagation} \tag{1}$$

In summary, the control plane latency is reduced by: (a) scheduler design, (b) speculative transmission, (c) switch topology and (d) switch routing. The average end-to-end latency is further reduced by having: (a) wavelength-striped packets and (b) virtual output queuing (VOQ) at the switch inputs. The focus of this work is the scheduler design, optimized for clock speed and scalability.

IV. Scheduler Design

The switch architecture or topology as well as the routing scheme directly affect the scheduler design and therefore the scheduling delay, $t_{scheduler}$. In this work we design the scheduler for a Clos network [30] used as the switch architecture and apply a fixed-path routing scheme, to reduce the scheduler delay and also simplify its design.

A Clos network is built using strictly non-blocking switching modules arranged in a multi-stage topology. A 3-stage Clos network is characterized by the triple (m, n, r), where m is the number of central modules (CMs), n is the number of input (output) ports on each input (output) module and r is the number of input modules (IMs) and output modules (OMs), as shown in Fig. 3. In an $N \times N$ Clos switch, $N = n \times r$. The

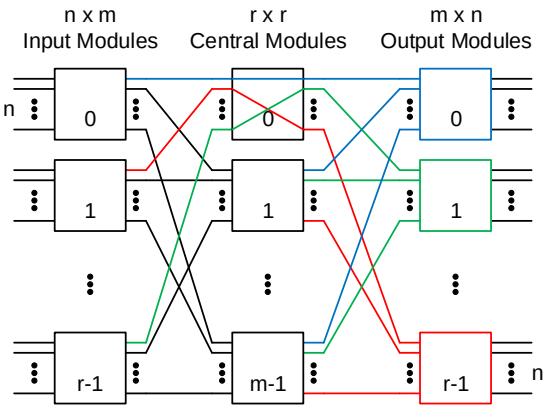

Fig. 3. An (m,n,r) Clos network built with strictly non-blocking modules. The proposed routing scheme, for m = n = r Clos networks, assigns fixed paths to avoid contention at the central modules, simplifying the scheduler design and reducing its delay.

case where $m = n = r = \sqrt{N}$, is attractive for photonic integration [31] and it gives a re-arrangeably non-blocking network, given an appropriate routing algorithm is used. More importantly, the modular structure of Clos networks enables reducing the scheduler delay because only $\sqrt{N}$-bit arbitration circuitry would be needed per module, for distributed path allocation.

Any multi-stage architecture, such as the Clos network, requires a routing algorithm. In the Clos network there are m paths from any input port, i, to any output port, j. The *looping* algorithm [38] can be used for Clos networks with $m = n$; it iterates the routing matrix, re-arranging current entries to add new routes, without blocking others. Iterating would cause a long scheduling delay, limiting scalability.

In this design, the routing algorithm assigns fixed paths to minimize the routing overhead. Furthermore, it completely eliminates contention at the central modules, as shown in Fig. 3. Since there is no need to arbitrate for any of the central modules, the scheduler design is simplified, less hardware resources are required and delay is further reduced. The trade off is that the architecture becomes blocking; at every IM at most one input port can be allocated the route to an OM, even if different output ports on that OM are requested. Hence, the switch throughput will saturate at a lower input traffic load compared to using the looping algorithm, for a given switch size. Switch average throughput measurements are presented in section VI.

The fixed path assignment could also be used to simplify the switch design. Since there is no switching activity at the central modules, these can be removed resulting in a 33% reduction in cross-points and 50% reduction in waveguides, improving signal integrity and reducing the power requirement. This makes the switch more practical to implement as a photonic-integrated circuit.

Figure 3 shows how the routing algorithm assigns the first path (CM 0), at every IM. More specifically, at every IM, the first path/output port leads to a different OM to avoid

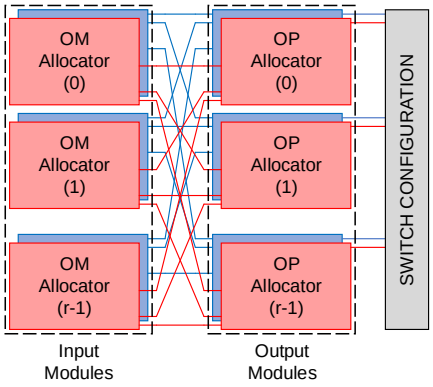

Fig. 4. Central Clos switch scheduler implemented in a planar and modular fashion. Planes operate independently and in parallel to allocate paths based on output module (OM) and output port (OP) arbitration for the Clos input and output modules. The switch configuration module reconfigures the switch based on the path allocation results from both planes.

contention at CM 0. This is the case for the remaining paths assigned. At any given IM, the path assignment to the OMs is a circular shift by one position to the left, with respect to the previous IM. It is calculated based on the input port (i) and destination output port (j) associated with every packet, according to the following equation:

$$p(i,j) = (\lfloor i/n \rfloor + \lfloor j/r \rfloor) \bmod m \qquad (2)$$

where $0 \leq i, j \leq N-1$.

Since there can be no contention at the central modules, path allocation is implemented as output port arbitration for every input and output module. This translates to allocating OMs to the IM input ports and then allocating OM output ports to the IMs.

The central scheduler is implemented in a modular planar approach as illustrated in Fig. 4. This allowed optimisation for clock speed. There are two parallel path-allocation planes, one responsible for processing new packet (server) requests and one for processing VOQ packet (switch) requests. On each plane, there are r allocation modules for the Clos IMs and another r allocation modules for the Clos OMs, arranged in two ranks and interconnected in a full mesh topology. The switch configuration module processes the allocation results from both planes to produce the configuration control signals for the Clos input and output modules. Every allocation module or *allocator*, irrespective of rank and plane, makes decisions based on a group of *round-robin* arbiters, where every arbiter is designed as outlined in [39]. In the next sections, we present the output module (OM) allocator design for each plane, the output port (OP) allocator design common to both planes and the switch (re)configuration design.

A. Output Module Allocation for New Packets

The allocator design for an $n \times r$ IM, for new packet requests, is shown in Fig. 5. For every new packet arriving at an input port, there is a switch output port request based on the packet's destination. The circuit processes every request

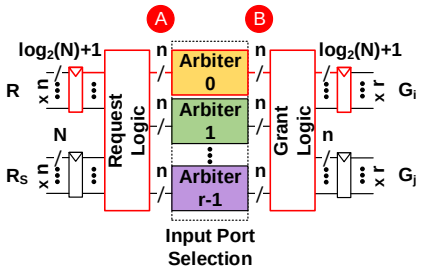

Fig. 5. Output module allocator design for new packet requests. The critical path in the design is shown in red. Tags A and B are for cross-reference with the example in Fig. 6.

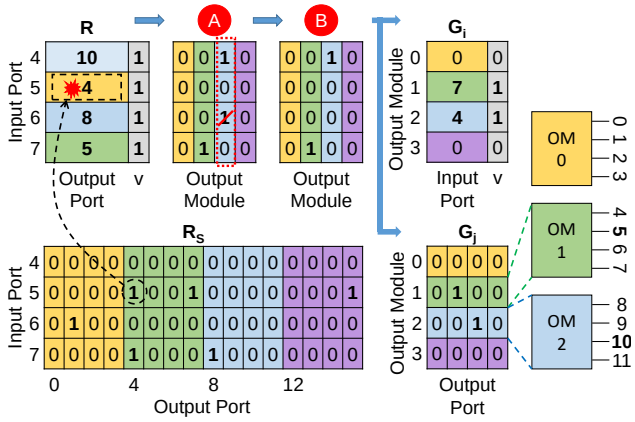

Fig. 6. Example request processing for new packets by the output module allocator for the 2nd input module in a (4,4,4) Clos switch. Binary matrices are tagged for cross-reference with the digital design in Fig. 5.

and attempts to match each input port to the OM on which the requested output port physically resides. There are two sets of input requests to the circuit: (a) n requests for the new packets, in matrix R, and (b) n requests for VOQ packets at the IM input ports, in matrix R_S. The VOQ requests are high-priority and are used here only to filter out new packet requests that they are contending against for IM input ports. This maintains strict in-order packet delivery. For the filtered out requests, the corresponding packets arriving at the switch are stored in the appropriate VOQs.

Every request in R is a structure with two fields; the switch destination port, represented by $log_2 N$ bits, and a valid bit. Every request in R_S is an N-bit vector, where every bit asserted represents a request for the output port that the VOQ buffers packets.

The request logic checks R_S to determine whether a new packet contends with a VOQ packet for an IM input port, in which case the new packet request is dropped. Then, it generates a request in a format that the subsequent arbiter logic can process; based on a new packet's destination port an n-bit request for the destination OM is generated, where each bit asserted is an IM input port requesting that OM. There are r arbiters, each allocating an OM to at most one of n requests, based on the round-robin principle. Each arbiter outputs a one-hot n-bit grant indicating the winning IM input port.

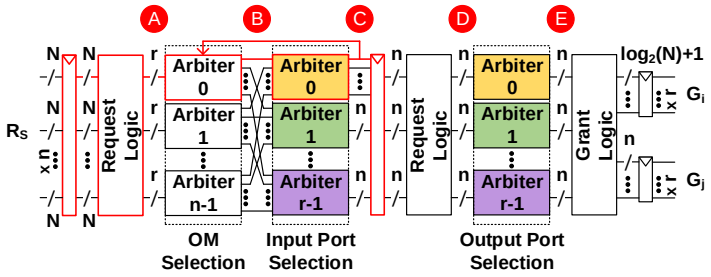

Fig. 7. Output module allocator design for switch VOQ packet requests. The critical path in the design is shown in red. Tags A-E are for cross-reference with the example in Fig. 8.

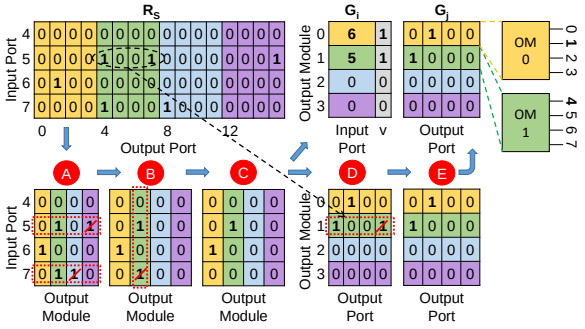

Fig. 8. Example request processing for switch VOQ packets by the output module allocator for the 2nd input module in a (4,4,4) Clos switch. Binary matrices are tagged for cross-reference with the digital design in Fig. 7.

The grant logic generates the circuit's output matrices, G_i and G_j, based on the arbiter grants. Each matrix has at most r grants, one for every Clos OM. Every grant in G_i is a structure with an input-port field of $log_2 N$ bits and a valid bit, to indicate the switch input port granted that OM. Every G_j grant is a one-hot n-bit vector and the bit asserted, if any, indicates the granted output port on that OM. Every valid grant pair, one from G_i and one from G_j, forms a request that the dedicated output port (OP) allocator on the same plane will process next, as illustrated in Fig. 4.

Figure 6 shows an example of the circuit functionality for a $(4, 4, 4)$ Clos, for a given R and R_S. The requests processed in the example are for the second input module (IM 1) whose input ports are in the range 4 to 7. The request logic is shown generating a binary matrix containing the arbiter requests, ignoring input port 5's request as it contends with a VOQ request for the same input and output ports. Next, every arbiter resolves any OM contention by operating across a matrix column, selecting only one input port for that OM. The output matrices G_i and G_j are created based on the arbitration results.

B. Output Module Allocation for Switch VOQ Packets

In an $n \times r$ IM, due to VOQ buffering, there could be up to N switch output port requests per input port, at the same time. For every input port, the OM allocator grants at most one of r OMs, for a *selected* output port. This design is only used in the scheduler plane responsible for VOQ request processing, as illustrated in Fig. 4. As shown in Fig. 7, the design is divided in two pipeline stages to increase the maximum clock frequency achievable.

Every input request in R_S is an N-bit vector. This is the same input matrix in the allocator for new packet requests described above. In the first pipeline stage, the request logic generates the arbiter input requests; n r-bit vectors in which every bit asserted is a request for an OM.

Next, input-first separable allocation is performed to match output modules with the local input ports. Separable allocation is performed as two sets of arbitration; one across the input ports and one across the output modules. This is implemented as two separate arbiter ranks. The first rank has n r-bit arbiters to select one OM for every input port. The second rank has r n-bit arbiters to select one input port for every OM.

The iSLIP method [40], for round-robin separable allocation, is used to increase the number of grants. This is done by controlling the arbiter priority in the first rank so that it is less likely different arbiters select the same output module, causing fewer conflicts in the second arbiter rank. Although multiple iterations can be performed, to further increase the number of grants and therefore the switch throughput, only a single pass (1-SLIP) is executed to minimize the total circuit delay. To implement iSLIP, the priority of a first-rank arbiter is updated only if its grant output has also won arbitration in the second arbiter rank. However, this creates a feedback which forms the critical path in the design, as discussed in section V. Any first-rank arbiter receives r feedback signals in total, one from each second-rank arbiter, but only the feedback between the first two arbiters is shown in in Fig. 7, for diagram clarity.

In the case of switch VOQ packets, since there can be more than one output port request per OM, an additional round of arbitration is needed to select one, after an input port has been granted an OM. This is implemented in the second pipeline stage. Based on the OM allocation grants from the first pipeline stage, the request logic creates n-bit requests, for every granted OM, and the following arbiters select one. Effectively the two allocation rounds operate in a master-slave fashion divided in two pipeline stages.

In the same way as in the allocator for new packet requests, the grant logic generates two matrices, G_i and G_j, each holding a grant for every Clos output module that the corresponding output port (OP) allocator will process.

An example of the circuit functionality is shown in Fig. 8.

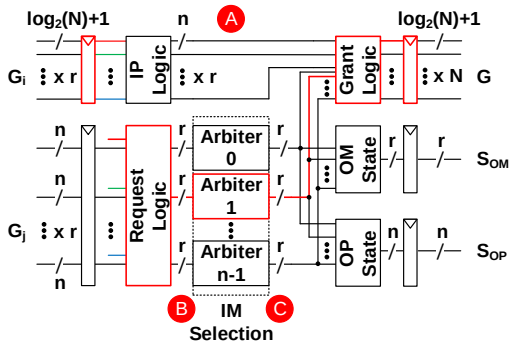

Fig. 9. Output port allocator design. The same design is used for new packet requests and for switch VOQ packet requests. The critical path in the design is shown in red. Tags A-C are for cross-reference with the example in Fig. 10.

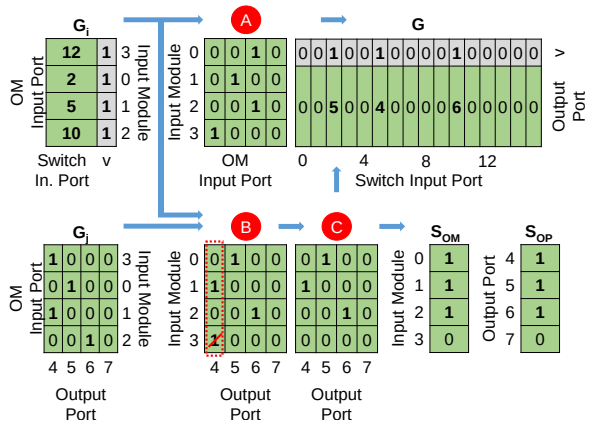

Fig. 10. Example request processing by the output port allocator for the 2nd output module in a (4,4,4) Clos switch. Binary matrices are tagged for cross-reference with the digital design in Fig. 9.

It extends on the previous example in Fig. 6 showing how the R_S matrix is processed for the same input module. First, the request logic creates the input port-OM binary matrix. Next, during separable allocation, the first arbiter rank operates across the matrix rows to select an OM for every input port and then the second arbiter rank operates across the columns to select one input port for every OM. The G_i matrix is created based on the allocation results, indicating input ports 5 and 6 have been granted output modules 1 and 0 respectively. An additional allocation round is executed before G_j is created to select one output port per granted OM. For input port 5, the first port on OM 1 (output port 4) is selected.

C. Output Port Allocation

The output port (OP) allocator design for an $r \times n$ OM is shown in Fig. 9. The circuit receives one G_i grant and one G_j grant from every OM allocator on the same plane, to form an IM request for a local output port. It then attempts to grant every local output port to at most one input module. The same design is used in both scheduler planes shown in Fig. 4.

The input port (IP) logic, for every valid G_i grant, reads the switch input port and locates it on the associated IM. It generates r n-bit vectors, one for every IM, and the bit asserted is the input port that has a request for this OM. This information enables the grant logic to group its output signals per IM.

The request logic uses G_i and G_j to create a binary matrix containing the arbiter requests. The $n \times r$ matrix indicates the output port every IM is requesting. Next, the arbiters select one OM output port for every IM, resolving any output port contention between IMs.

Once an arbiter grants an output port to an IM, the grant logic then matches it to the input port on that IM. This is done by pairing the outputs of the IP logic with the corresponding arbiter grants. Also, the OM output port is translated into a switch output port j, where $0 \leq j \leq N-1$. There are N output grants in G, one for each input port, grouped into IMs and every grant is a structure with a valid bit and an output port field of $log_2 N$ bits.

The logic for the output module (OM) state creates an r-bit vector S_{OM} based on the arbiter grants to mark every IM granted an output port. The logic for the output port (OP) state creates an n-bit vector S_{OP} to mark every granted output port. Both S_{OM} and S_{OP} will be used in the switch configuration circuit to filter out new packet grants contending with VOQ packet grants for an output module or an output port.

Figure 10 illustrates an example of the circuit's functionality for OM 1, interfacing with the OM allocators for switch VOQ packets. This includes the G_i grant and G_j grant from IM 1 to OM 1, shown in Fig. 8, which collectively form a request for output port 4 from input port 5. The input port logic first creates a binary matrix to indicate the input port on every IM that has a request for this output module. Next, the request logic creates the arbitration binary matrix and every arbiter operates across a matrix column selecting at most one IM for every output port. Based on the arbitration results, the G matrix holds the input-output port matches and the S_{OM} and S_{OP} vectors indicate the winning IMs and allocated ports. In this example, input ports 2, 5 and 10 have been granted output ports 5, 4 and 6, respectively.

D. Switch Configuration

The switch configuration circuit generates the input and output module reconfiguration signals, C_{IM} and C_{OM}, which switch on the appropriate optical gates based on the routing scheme. It also generates the write-enable (C_{WE}) and read-enable (C_{RE}) signals to store and release packets at the VOQs. The design is shown in Fig. 11, using only the OP allocator outputs from the two planes, as shown in Fig. 4. The states of the output ports and output modules are only taken from the OP allocators in the VOQ plane to filter out paths allocated in the other plane, as explained below. We use G and G_S to distinguish the allocator grant matrices from each plane.

The grant matrix G_S contains the OP allocator grants in the VOQ plane and it is distributed to all logic blocks in this circuit. It is used solely in the read-enable block and in one of the IM and OM configuration blocks, to produce a switch configuration in response to VOQ requests only. In the remaining blocks, together with S_{OM} and S_{OP}, it is used to filter G which is then used to produce a switch configuration in response to new packet requests.

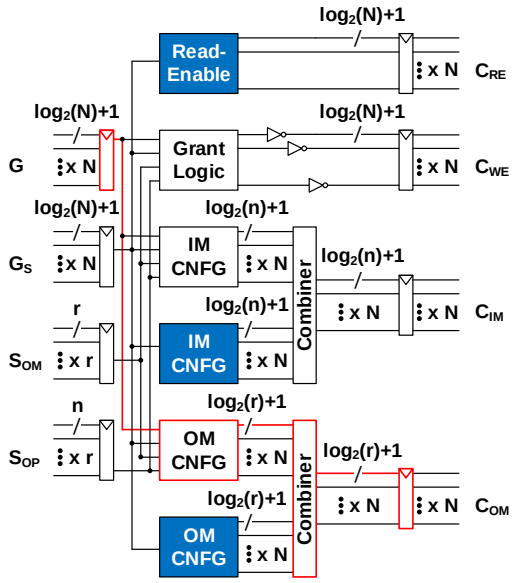

Fig. 11. Switch configuration circuit. Blue-coded logic blocks are for VOQ requests and otherwise for new requests. The critical path in the design is shown in red.

The read-enable logic transfers G_S directly to the output register for the VOQ read-enable signals in C_{RE}. In total there are N signals, one for every switch input port, where each signal is a structure with a valid bit and $log_2(N)$ bits to address each VOQ. A valid read-enable signal releases a packet from the corresponding switch VOQ at an input port.

The IM and OM configuration logic blocks for VOQ requests, shown in blue color in Fig. 11, generate a configuration signal for every valid grant in G_S. An IM configuration signal uses $log_2(n)$ bits to identify one out of n optical gates on an IM output port to turn on. Similarly, an OM configuration signal uses $log_2(r)$ bits to identify one out of r optical gates on an OM output port to turn on.

At the grant logic, every valid grant in G is passed to the output only if: (a) no G_S grant is valid for that IM input port, (b) the destination OM is free for that IM, according to S_{OM} and (c) the destination output port j is free, according to S_{OP}. This 3-condition filter gives priority to VOQ packets. Any grant passing through the filter corresponds to a new packet being switched with the minimum latency. The outputs are inverted and registered to produce the write-enable signals in C_{WE}, where a valid signal stores a new packet in the switch VOQ addressed by $log_2(N)$ bits.

The IM and OM configuration logic for new packets generates a configuration signal for every valid grant in G, given that grant also passes through the same 3-condition filter used in the grant logic. The IM configuration signals for new packets and VOQ packets are then combined together into a single set of N signals before they are registered out as C_{IM}. In the same way, the output OM configuration, C_{OM}, is generated by combining the configuration signals for new packets and switch VOQ packets.

TABLE II
MINIMUM CLOCK PERIOD FOR ASIC SYNTHESIS

Hardware Module	Switch Size		
	16x16	64x64	256x256
Output Module Allocator	0.8 ns	1.1 ns	1.5 ns
Output Module Allocator (iSLIP)	1.1 ns	1.4 ns	2.0 ns
Output Port Allocator	0.8 ns	1.1 ns	1.5 ns
Switch Configuration	0.5 ns	0.8 ns	1.1 ns

V. SCHEDULER IMPLEMENTATION RESULTS

In this section we discuss the implementation of the modular scheduler design on hardware. More specifically, every hardware module described in the previous section has been implemented separately as an application-specific integrated circuit (ASIC), in a 45 nm CMOS process. The critical path in each module is identified and measured to determine the minimum clock period that the scheduler, as a whole unit, can achieve. The modules are also implemented on a Virtex-7 XC7V690T FPGA board to investigate how the scheduler prototype compares to our previous work and to related work in the literature. Carry-look-ahead was used for the round-robin arbiters in the allocator modules for the 64-port and 256-port scheduler, both in the FPGA and ASIC implementation.

A. Application-Specific Integrated Circuit (ASIC)

The critical path, for every scheduler component, is shown in its circuit diagram in section IV. Table II lists the corresponding minimum clock period for the ASIC implementation. The longest critical path is in the output module allocator for switch VOQ packets. It falls in the first pipeline stage responsible for the iSLIP allocation. It extends from the request register through a cascade of two iSLIP arbiters and then back to the priority register of the first arbiter, as shown in Fig. 7. This sets the minimum clock period, $T_{scheduler}$, for the entire scheduler at 2 ns for a 256×256 Clos switch.

According to Table II, the increase in minimum clock period, as the switch quadruples in size, is sublinear for all scheduler modules, demonstrating a scalable scheduler design. The OM allocator for new packets and the OP allocator use same size arbiters, request and grant logic, hence the critical path is of the same length. Moreover, there is only a small difference in minimum clock period between the different scheduler components, for the same switch size, indicating an overall pipelining balance in the scheduler design. As we reported in [29], the entire scheduler can be implemented as a 3-stage pipelined circuit, in which case its total delay, for a 256×256 switch, would be $t_{scheduler} = 3 \times T_{scheduler} = 6$ ns. In a synchronous rack-scale (2 m) system, the total control plane delay would be less than 20 ns, including request generation (2 ns), propagation to scheduler (10 ns), scheduling (6 ns) and SOA-based switch reconfiguration (0.115 ns [41]).

B. Field-Programmable Gate Array (FPGA)

The iSLIP output module allocator was implemented on the Xilinx Virtex-7 XC7V690T FPGA board to compare the

TABLE III
MINIMUM CLOCK PERIOD FOR FPGA IMPLEMENTATION

Allocation Design	Switch Size			
	16x16	32x32	64x64	256x256
OM Allocator (iSLIP)	2.9 ns	-	4.5 ns	7.0 ns
Andreades et. al [28]	-	5.4 ns	-	-

scheduler delay to ASIC implementation, previous work and also to related work in the field. The critical path in the allocator is the same as in the ASIC implementation.

Table III lists the minimum clock period for different switch sizes. As the switch quadruples in size, the minimum clock period again shows a sublinear increase, which verifies scheduler scalability. Compared to the ASIC OM allocator in Table II, the scheduler would run 2.6 to 3.5 times slower on the FPGA board as the switch radix is increased from 16 to 256, respectively. However, against the Clos scheduler design in previous work [28], a switch twice the size can now be scheduled 16.7% faster. This is due to distributed path allocation and routing scheme. The total scheduler delay, compared to ASIC implementation and other related work, is listed in Table I in section II.

VI. NETWORK EMULATION RESULTS

A network emulator was built to evaluate the switch performance under the control of the scheduler in our system concept, described in section III. The emulator was developed in *SystemVerilog*, a hardware description language (HDL) for digital design and hardware implementation, and ran in Mentor Graphics *ModelSim*. Using an HDL emulator allows for cycle-accurate measurements. The emulator consists of N packet sources, the network model and the packet sink, arranged as shown in the block diagram in Fig. 12.

Each packet source injects traffic with a uniform random packet inter-arrival and the injection probability is set by a universal load parameter. The packets are timestamped in the clock cycle they are generated. Also, the packet destinations (switch output ports) are random and uniformly distributed.

The network model implements an $N \times N$ optical switch, the switch scheduler, N server network interfaces each with a single FIFO queue and another N network interfaces for the switch input ports each with N FIFO virtual output queues (VOQs). Every packet source feeds one server network interface. The packet sink receives all output packets from the switch instance, adds a second timestamp to each packet and performs all necessary measurements.

The emulation captures the end-to-end latency in our system concept, shown in Fig. 2. It includes control plane overheads for request control (T_{TX}), transport ($t_{propagation}$) and scheduling ($t_{scheduler}$) as well as data plane overheads for packet buffering, transport ($t_{propagation}$), de-serialization (t_{serial}) and switch reconfiguration (t_{switch}).

The emulation parameters are listed in Table IV. The (m, n, r) Clos configurations considered for the switch architecture are the (4,4,4), (8,8,8) and (16,16,16) for sizes 16×16, 64×64 and 256×256. Built using SOA technology,

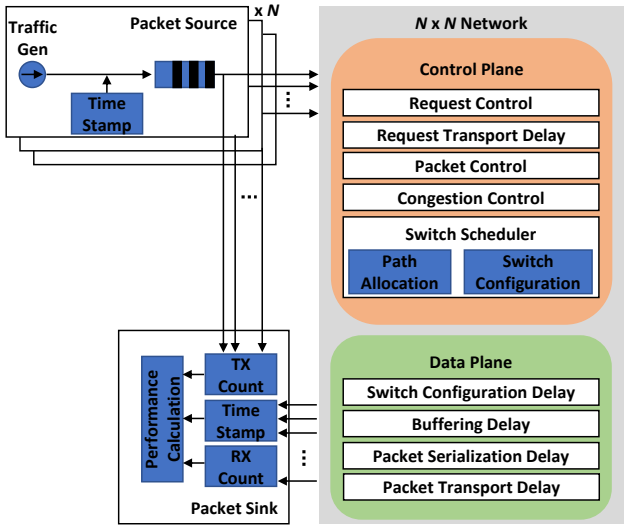

Fig. 12. The network emulation setup.

TABLE IV
NETWORK EMULATION PARAMETERS

Parameter	Value
Measurements Duration	20 μs
Switch Size - N	16, 64, 256
$T_{scheduler}$ (ASIC)	1.1 ns, 1.4 ns, 2.0 ns
T_{TX}	$T_{scheduler}$
t_{switch} (SOA Rise Time)	0.115 ns
t_{serial} ($k\times$ 100 Gb/s)	$T_{scheduler}$ - t_{switch}
$t_{propagation}$ (2 m)	10 ns

the reconfiguration time, t_{switch}, can be as low as 115 ps [41]. The clock period, T_{TX}, at which the packet sources and server network interfaces are running, is equal to the scheduler clock period, $T_{scheduler}$, for a synchronous system. The packets are assumed to be wavelength-striped across k wavelengths each running at 100 Gb/s, with a de-serialization latency, $t_{serial} \leq T_{scheduler} - t_{switch}$. The propagation delay ($t_{propagation}$) is set to 10 ns to model a rack-scale system with 2 m link distances.

Figure 13 shows the average end-to-end latency against the input traffic load, as a percentage of capacity, for different size Clos switches and ASIC scheduler. At low loads, the average end-to-end latency for each switch size converges to the minimum value; it consists of 1 T_{TX} buffer control delay at the packet source and server network interface, request propagation ($t_{propagation}$) to the scheduler, 3 $T_{scheduler}$ scheduling, switch reconfiguration (t_{switch}), packet propagation ($t_{propagation}$) to the receiver and finally packet de-serialization (t_{serial}) at the receiver. The minimum latency is longer for larger switches, due to longer $T_{scheduler}$, and is given by the following equation:

$$t = 2T_{TX} + 2t_{propagation} + t_{scheduler} + t_{switch} + t_{serial} \quad (3)$$

where $t_{scheduler} = 3 \times T_{scheduler}$. For the 256-port switch, $t = 32.0$ ns out of which 20 ns is due to 2×10 m propagation. Clock and data recovery (CDR) at the receiver is excluded but

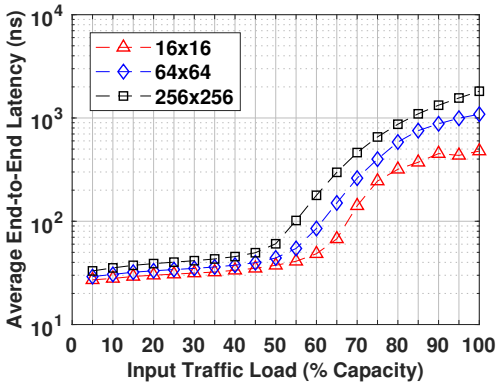

Fig. 13. Average end-to-end latency vs. input traffic load, for different size Clos switches. Results are based on scheduler implementation on 45 nm CMOS ASIC. The network emulation assumes uniform random traffic and wavelength-striped packets of at most 1 clock cycle duration. A 2 m fibre distance to and from the switch is assumed to model rack-scale switching.

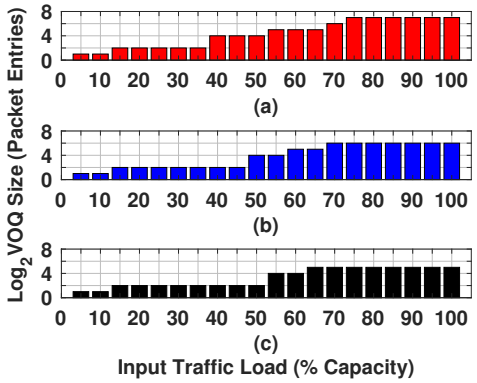

Fig. 14. Minimum switch VOQ size vs. input traffic load, for (a) 16 x 16, (b) 64 x 64 and (c) 256 x 256 Clos switch sizes. Results are based on scheduler implementation on 45 nm CMOS ASIC. The network emulation assumes uniform random traffic and wavelength-striped packets of at most 1 clock cycle duration.

the overhead will not be significant, given the sub-nanosecond CDR time reported in [42].

For all switch sizes, as the traffic load is increased, the contention probability in the Clos switch increases and more packets are buffered at the switch, which in turn increases the packet average queuing delay and hence the end-to-end latency. Also, for a given traffic load, the contention probability is higher for larger switches, resulting in longer latency, which is more pronounced at higher traffic loads. For example, for latency less than 100 ns, larger switches need to be operated at a lower traffic load. Nevertheless, the decrease is only 5% per $4\times$ increase in switch size, indicating scalability.

Next we examine the buffering resources required for the latency performance in Fig. 13. For the server network interface a small 4-packet buffer is enough to support input traffic at 100% of capacity, irrespective of switch size. This is attributed to speculative transmission due to which a packet experiences a fixed delay of 4 clock cycles in total; 1 clock cycle to be registered and 3 more until the pipelined scheduler processes the request ($t_{scheduler} = 3 \times T_{scheduler}$).

On the other hand, the switch VOQ size required varies with the input traffic load and switch size. Figure 14 shows the minimum VOQ size in packet entries, required at the switch to avoid packet loss, as a function of the input traffic load. As expected, the size requirement increases with increasing load due to higher contention and therefore more packets would need to be buffered at the switch. At high loads, the size per VOQ decreases with the switch size because the probability of a destination occurring is lower, under uniform random traffic. For the 256 x 256 switch, the minimum VOQ size at 100% load is 32 packets, which translates to 512 KB per switch input port for 64 B packets. Using control backpressure for buffering management, a small 8-packet VOQ would suffice for all switch sizes and traffic loads, in this rack-scale emulation, at the cost of a larger buffer at the server network interface.

The cumulative distribution of the packet end-to-end latency for different size Clos switches, at 25%, 50% and 75% input traffic loads, is shown in Fig. 15. These are the distributions from which the mean/average latency values are calculated and plotted in Fig. 13. At 25% of the input traffic capacity, more than 20% of the packets are received with the minimum latency. At higher loads, in addition to path contention with other servers, new packets experience an increased contention with the packets already buffered at the switch VOQs, which have a higher priority. This trades off the minimum latency packet proportion with reduced average latency at higher loads, as shown in Fig. 13, and also maintains a strict in-order packet delivery. The insets in Fig. 15 focus on the distribution tails, accounting for 99% to 100% of the received packets. The tails have a latency range of **less than an order of magnitude** for all switch sizes even at a 75% input traffic load.

The average latency for the 256×256 switch, for the FPGA-based scheduler, is shown in Fig. 16. It is calculated based on the $T_{scheduler}$ value listed in Table III. Compared to our previous 32×32 switch design [28], sub-microsecond latency is now achieved up to a 65% load; a 30% gain for a switch size eight times larger. Moreover, at 65% input traffic load, there is now an order of magnitude decrease in average latency. The performance improvement is due to: (a) distributed path allocation, (b) fixed and reduced-conflict routing scheme and (c) virtual output queuing at the switch inputs.

Figure 17 shows the switch average throughput, measured in packets per clock cycle, against the input traffic load. At 100% input traffic load, an $N \times N$ switch receives N packets per clock cycle. The average throughput is expressed as a percentage of output capacity ($= N$) to examine the performance as the switch size is increased. The results show that the switch average throughput saturates at 60% of the input traffic capacity. The capacity in our system can be very high due to the wavelength-striped transmission format. Furthermore, there is a very small penalty in throughput performance every time the switch quadruples in size, demonstrating good scalability. In comparison to the 32×32 switch in previous work, the current design increases the saturation throughput by 28% and that for a 256×256 switch.

The scheduler fairness, in terms of switch output port allocation to the input ports, may be assessed by measuring the average latency variation at the flow-level. A packet flow refers

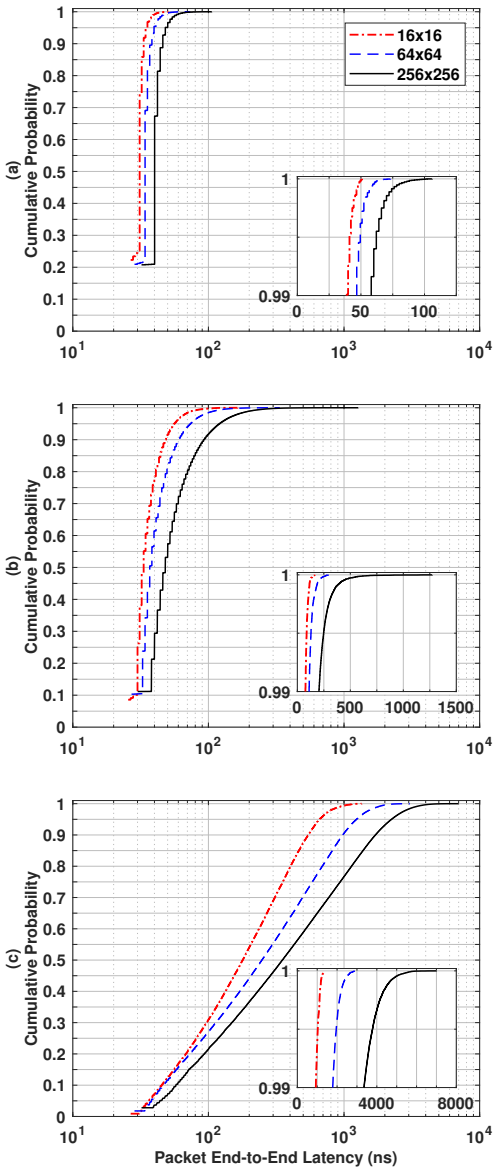

Fig. 15. Cumulative distribution of the packet end-to-end latency at (a) 25%, (b) 50% and (c) 75% input traffic loads, for different size Clos switches. Results are based on scheduler implementation on 45 nm CMOS ASIC. The network emulation assumes uniform random traffic and wavelength-striped packets of at most 1 clock cycle duration. A 2 m fibre distance to and from the switch is assumed to model rack-scale switching.

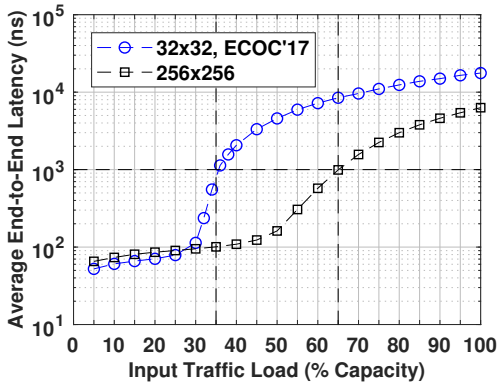

Fig. 16. Average end-to-end latency vs. input traffic load, for different Clos switch schedulers. Results are based on scheduler implementation on the Virtex-7 XC7V690T FPGA board. The network emulation assumes uniform random traffic and wavelength-striped packets of at most 1 clock cycle duration. A 2 m fibre distance to and from the switch is assumed to model rack-scale switching.

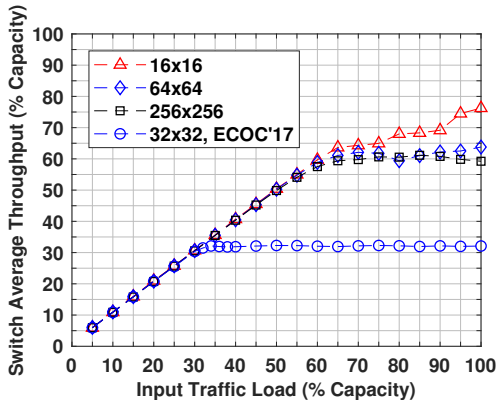

Fig. 17. Switch average throughput vs. input traffic load, for different size Clos switches. Throughput is measured as the total number of packets per clock cycle across all output ports. Point-to-point connections are used as a throughput reference in a system with no contention.

to packets from one switch input port to the same output port. Therefore, in an $N \times N$ switch, there are N flows per output port. Figure 18 shows the standard deviation of the N flow average latencies, for each output port, at different input traffic loads. The standard deviation is shown increasing with switch size and traffic load, indicating a decrease in fairness. Although round-robin arbitration is fair locally, at the system level the scheduler is less fair. In general, there is an inherent fairness penalty in distributed arbitration scheduling, where resource access is decided for only a subset of requests. Furthermore, parallel-plane path allocation for new packets and higher-priority VOQ packets drops an increasing proportion of new packet grants with traffic load, trading off fairness for strict in-order delivery and low average latency.

VII. Conclusion

We presented a highly-parallel and modular scheduler design for optical Clos switches which are practical for scalable photonic-integrated packet switching. A novel fixed path allocation scheme was used to completely eliminate contention at the Clos central modules. For the input and output modules, allocation is distributed to dedicated scheduler modules to resolve any contention. The design for each scheduler hardware module has been optimized for clock speed, using a parallel and pipelined implementation, and was presented and discussed in detail.

Each hardware module in the scheduler was synthesized as an ASIC in a 45 nm CMOS process. The implementation results show that each module's minimum clock period scales sublinearly with the switch size, verifying scalability. Furthermore, for a 256×256 Clos switch, the minimum clock period for the entire scheduler unit is 2.0 ns, limited by iSLIP allocation for the Clos input modules. This results

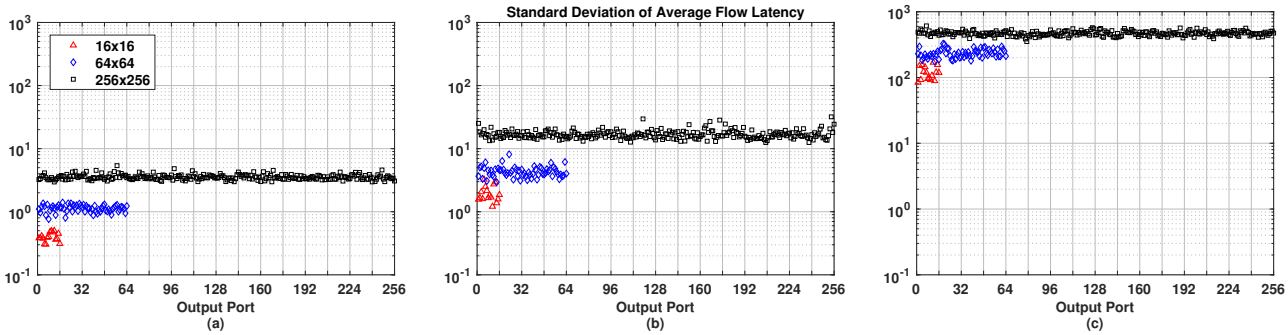

Fig. 18. Standard deviation of the average flow latency per output port at (a) 25%, (b) 50% and (c) 75% input traffic loads, for different size Clos switches. Results are based on scheduler implementation on 45 nm CMOS ASIC. The network emulation assumes uniform random traffic and wavelength-striped packets of at most 1 clock cycle duration. A 2 m fibre distance to and from the switch is assumed to model rack-scale switching.

in a total scheduling delay of only 6.0 ns, in a 3-stage pipelined implementation of the scheduler, outperforming the fastest designs in the literature for the switch sizes considered here.

A cycle-accurate rack-scale emulation of our switch system concept was developed to measure the Clos switch performance under uniform random traffic. Based on the ASIC implementation results, the minimum end-to-end latency for a 256×256 size is 32.0 ns, out of which only 6.0 ns are attributed to central scheduling. For this switch size, the average end-to-end latency remains on nanosecond time scales up to 80% of input traffic load. The distribution of the packet latency shows a short tail for all switch sizes examined, **within an order of magnitude** from the 99th percentile, even at a 75% load.

For implementation on the Xilinx Virtex-7 XC7V690T FPGA board, the 256-port switch average latency performance remains on nanosecond timescale up to a 65% traffic load. Compared to a 32-port Clos switch in previous work, this is a 30% load improvement for a switch size eight times larger. The performance gain is due to routing scheme, distributed path allocation and virtual output queuing (VOQ) at the switch.

The switch achieves zero packet loss for all sizes up to 100% of input traffic load. This is enabled by VOQ buffering at the switch input ports. For the 256-port switch, the VOQ size requirement is 32 packets to avoid packet loss at full port capacity, but can be as short as 8 packets long if control backpressure from the switch to the servers is applied.

The average throughput was also measured for all switch sizes and compared to previous work. The results showed that the switch saturates near 60% of port capacity with a very small penalty as the switch scales in size. For all sizes, the switch maintains the peak throughput beyond saturation as the load is increased, indicating that the switch is stable. Moreover, compared to the 32-port Clos switch in previous work, the throughput has now doubled for a switch size 8 times larger.

In distributed arbitration scheduling, resource allocation fairness is degraded as decisions are based only on a subset of requests. In this work, scheduler fairness was assessed by examining the packet flow average latency from different switch input ports to the same output port. Our results showed a switch size and traffic load dependency on the fairness degradation. Future work will investigate scheduler design techniques to mitigate this.

ACKNOWLEDGMENT

This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R035342/1 and in part by the EU Horizon 2020 programme (Industrial Leadership section) under Grant 687632.

REFERENCES

[1] Cisco Systems, Inc., "Cisco Global Cloud Index: Forecast and Methodology, 2016 - 2021," Cisco Systems, Inc., White Paper, 2018.
[2] G. Lee, *Cloud Networking: Understanding Cloud-based Data Center Networks*. Morgan Kaufmann, 2014.
[3] P. Goransson and C. Black, *Software Defined Networks: A Comprehensive Approach*. Morgan Kaufmann, 2014.
[4] N. Chrysos, F. Neeser, M. Gusat, C. Minkenberg, W. Denzel, C. Basso, M. Rudquist, K. M. Valk, and B. Vanderpool, "Large switches or blocking multi-stage networks? An evaluation of routing strategies for datacenter fabrics," *Computer Networks*, vol. 91, pp. 316 – 328, 2015.
[5] M. Alizadeh and T. Edsall, "On the data path performance of leaf-spine datacenter fabrics," in *IEEE Hot Interconnects (HOTI)*, August 2013, pp. 71–74.
[6] Cisco Systems, Inc., "Cisco Nexus 3548 Switch Performance Validation," Cisco Systems, Inc., White Paper, December 2012.
[7] N. Zilberman, P. M. Watts, C. Rotsos, and A. W. Moore, "Reconfigurable network systems and software-defined networking," *Proceedings of the IEEE*, vol. 103, no. 7, pp. 1102–1124, July 2015.
[8] W. M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A. C. Snoeren, and G. Porter, "RotorNet: A scalable, low-complexity, optical datacenter network," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 267–280.
[9] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. E. Ng, M. Kozuch, and M. Ryan, "c-Through: Part-time optics in data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 327–338.
[10] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: ACM, 2010, pp. 339–350.
[11] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: A topology malleable data center network," in *ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, ser. HotNets-IX. New York, NY, USA: ACM, 2010, pp. 1–6.
[12] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat, "Integrating microsecond circuit switching into the data center," in *Proceedings of the ACM SIGCOMM 2013 Conference*, ser. SIGCOMM '13. New York, NY, USA: ACM, 2013, pp. 447–458.
[13] Q. Cheng, L. Y. Dai, N. C. Abrams, Y.-H. Hung, P. E. Morrissey, M. Glick, P. O'Brien, and K. Bergman, "Ultralow-crosstalk, strictly non-blocking microring-based optical switch," *Photonics Research*, vol. 7, no. 2, pp. 155–161, February 2019.

[14] P. DasMahapatra, R. Stabile, A. Rohit, and K. A. Williams, "Optical crosspoint matrix using broadband resonant switches," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 20, no. 4, pp. 1–10, July 2014.

[15] N. Calabretta, W. Miao, K. Mekonnen, K. Prifti, and K. Williams, "Monolithically integrated WDM cross-connect switch for high-performance optical data center networks," in *Optical Fiber Communications Conference and Exhibition (OFC)*, March 2017.

[16] Q. Cheng, A. Wonfor, J. Wei, R. V. Penty, and I. H. White, "Low-energy, high-performance lossless 8x8 SOA switch," in *Optical Fiber Communications Conference and Exhibition (OFC)*, March 2015.

[17] B. G. Lee, A. V. Rylyakov, W. M. J. Green, S. Assefa, C. W. Baks, R. Rimolo-Donadio, D. M. Kuchta, M. H. Khater, T. Barwicz, C. Reinholm, E. Kiewra, S. M. Shank, C. L. Schow, and Y. A. Vlasov, "Monolithic silicon integration of scaled photonic switch fabrics, CMOS logic, and device driver circuits," *Journal of Lightwave Technology*, vol. 32, no. 4, pp. 743–751, February 2014.

[18] N. Dupuis, F. Doany, R. A. Budd, L. Schares, C. W. Baks, D. M. Kuchta, T. Hirokawa, and B. G. Lee, "A nonblocking 4x4 Mach-Zehnder switch with integrated gain and nanosecond-scale reconfiguration time," in *Optical Fiber Communication Conference (OFC)*, 2019.

[19] M. Ding, A. Wonfor, Q. Cheng, R. V. Penty, and I. H. White, "Hybrid MZI-SOA InGaAs/InP photonic integrated switches," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 24, no. 1, pp. 1–8, January 2018.

[20] Z. Guo, L. Lu, L. Zhou, L. Shen, and J. Chen, "16x16 silicon optical switch based on dual-ring-assisted Mach-Zehnder interferometers," *Journal of Lightwave Technology*, vol. 36, no. 2, pp. 225–232, January 2018.

[21] Q. Yang, K. Bergman, G. D. Hughes, and F. G. Johnson, "WDM packet routing for high-capacity data networks," *Journal of Lightwave Technology*, vol. 19, no. 10, pp. 1420–1426, October 2001.

[22] A. Shacham and K. Bergman, "Building ultralow-latency interconnection networks using photonic integration," *IEEE Micro*, vol. 27, no. 4, pp. 6–20, July 2007.

[23] R. Luijten, C. Minkenberg, R. Hemenway, M. Sauer, and R. Grzybowski, "Viable opto-electronic HPC interconnect fabrics," in *ACM/IEEE Supercomputing Conference*, Nov 2005.

[24] R. Proietti, C. J. Nitta, Y. Yin, R. Yu, S. J. B. Yoo, and V. Akella, "Scalable and distributed contention resolution in AWGR-based data center switches using RSOA-based optical mutual exclusion," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, no. 2, pp. 3 600 111–3 600 111, March 2013.

[25] I. Cerutti, J. A. Corvera, S. M. Dumlao, R. Reyes, P. Castoldi, and N. Andriolli, "Simulation and FPGA-based implementation of iterative parallel schedulers for optical interconnection networks," *Journal of Optical Communications and Networking*, vol. 9, no. 4, pp. C76–C87, April 2017.

[26] P. Andreades, Y. Wang, J. Shen, S. Liu, and P. M. Watts, "Experimental demonstration of 75 ns end-to-end latency in an optical top-of-rack switch," in *Optical Fiber Communications Conference and Exhibition (OFC)*, March 2015.

[27] P. Andreades, K. Clark, P. M. Watts, and G. Zervas, "Experimental demonstration of an ultra-low latency control plane for optical packet switching in data center networks," *Optical Switching and Networking*, vol. 32, pp. 51–60, November 2019.

[28] P. Andreades and P. M. Watts, "Low latency parallel schedulers for photonic integrated optical switch architectures in data centre networks," in *European Conference on Optical Communication (ECOC)*, September 2017.

[29] P. Andreades and G. Zervas, "Parallel distributed schedulers for scalable photonic integrated packet switching," in *Photonics in Switching and Computing (PSC)*, September 2018.

[30] C. Clos, "A study of non-blocking switching networks," *The Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, March 1953.

[31] I. H. White, E. T. Aw, K. A. Williams, H. Wang, A. Wonfor, and R. V. Penty, "Scalable optical switches for computing applications [Invited]," *Journal of Optical Networking*, vol. 8, no. 2, pp. 215–224, Feb 2009.

[32] A. Shacham, B. A. Small, O. Liboiron-Ladouceur, J. P. Mack, and K. Bergman, "An ultra-low latency routing node for optical packet interconnection networks," in *IEEE Lasers and Electro-Optics Society (LEOS)*, vol. 2, November 2004, pp. 565–566.

[33] A. Shacham, B. G. Lee, and K. Bergman, "A wide-band nonblocking 2x2 switching node for a SPINet network," *IEEE Photonics Technology Letters*, vol. 17, no. 12, pp. 2742–2744, December 2005.

[34] J. Luo, S. Di Lucente, J. Ramirez, H. J. S. Dorren, and N. Calabretta, "Low latency and large port count optical packet switch with highly distributed control," in *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, March 2012.

[35] F. Yan, W. Miao, O. Raz, and N. Calabretta, "OPSquare: A flat DCN architecture based on flow-controlled optical packet switches," *Journal of Optical Communications and Networking*, vol. 9, no. 4, pp. 291–303, April 2017.

[36] W. Miao, J. Luo, S. Di Lucente, H. Dorren, and N. Calabretta, "Novel flat datacenter network architecture based on scalable and flow-controlled optical switch system," *Optics Express*, vol. 22, no. 3, pp. 2465–2472, February 2014.

[37] X. Zheng, D. Patil, J. Lexau, F. Liu, G. Li, H. Thacker, Y. Luo, I. Shubin, J. Li, J. Yao, P. Dong, D. Feng, M. Asghari, T. Pinguet, A. Mekis, P. Amberg, M. Dayringer, J. Gainsley, H. F. Moghadam, E. Alon, K. Raj, R. Ho, J. E. Cunningham, and A. V. Krishnamoorthy, "Ultra-efficient 10Gb/s hybrid integrated silicon photonic transmitter and receiver," *Optics Express*, vol. 19, no. 6, pp. 5172–5186, March 2011.

[38] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Elsevier, 2004.

[39] P. Gupta and N. McKeown, "Designing and implementing a fast crossbar scheduler," *IEEE Micro*, vol. 19, no. 1, pp. 20–28, January 1999.

[40] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, no. 2, pp. 188–201, April 1999.

[41] R. C. Figueiredo, N. S. Ribeiro, A. M. O. Ribeiro, C. M. Gallep, and E. Conforti, "Hundred-picoseconds electro-optical switching with semiconductor optical amplifiers using multi-impulse step injection current," *Journal of Lightwave Technology*, vol. 33, no. 1, pp. 69–77, January 2015.

[42] K. Clark, H. Ballani, P. Bayvel, D. Cletheroe, T. Gerard, I. Haller, K. Jozwik, K. Shi, B. Thomsen, P. Watts, H. Williams, G. Zervas, P. Costa, and Z. Liu, "Sub-nanosecond clock and data recovery in an optically-switched data centre network," in *European Conference on Optical Communication (ECOC)*, September 2018.