Optical Switching and Networking 32 (2019) 51–60

Contents lists available at ScienceDirect

Optical Switching and Networking

journal homepage: www.elsevier.com/locate/osn

Experimental demonstration of an ultra-low latency control plane for optical packet switching in data center networks

Paris Andreades [a,*], Kari Clark [a], Philip M. Watts [a,b], Georgios Zervas [a]

[a] Optical Networks Group, Department of Electronic and Electrical Engineering, University College London (UCL), London, WC1E 7JE, UK
[b] ARM Ltd., Cambridge, CB1 9NJ, UK

ARTICLE INFO

Keywords:
Physical-layer control plane design
Optical crossbar switch scheduling
Optical packet switching
Optical interconnects

ABSTRACT

Optical interconnection networks have the potential to reduce latency and power consumption while increasing the bisection bandwidth of data center networks compared to electrical network architectures. Optical circuit-switched networking has been proposed but it is reconfigurable in milliseconds. Although switches operating on nanosecond timescales have been demonstrated, centrally scheduling such switching architectures is considered to be of high complexity, incurring significant delay penalties on the total switching latency. In this paper we present a high-speed control plane design based on a central switch scheduler for nanosecond optical switching which significantly reduces the end-to-end latency in the network compared to using the best electronic switches. We discuss the implementation of our control plane on field-programmable gate array (FPGA) boards and quantify its delay components. We focus on the output-port allocation circuit design which limits the scheduling delay and the end-to-end latency. Using our FPGA-implemented control plane, for a 32×32 switch, we experimentally demonstrate rack-scale optical packet switching with a minimum end-to-end head-to-tail latency of 71.0 ns, outperforming current state-of-the-art electronic switches. The effect of asynchronous control plane operation on the switch performance is evaluated experimentally. Finally, a new parallel allocation circuit design is presented decreasing the scheduling delay by 42.7% and the minimum end-to-end latency to 54.6 ns. More importantly, it enables scaling to a switch double the size (64×64) with a minimum end-to-end latency less than 71.0 ns. In a developed cycle-accurate network emulator we demonstrate nanosecond switching up to 60% of port capacity and average end-to-end latency less than 10 μs at full capacity while maintaining zero packet loss across all traffic loads.

1. Introduction

Annual global data center traffic has been increasing by 27% every year since 2015 and it is projected that it will continue to do so reaching 15.3 zettabytes by the end of 2020, out of which 77% will be due to traffic within the data centers [1]. The traffic exchanged between servers within a data center is known as east-west traffic and it is driven by a number of technological trends. For example, the growth of cloud computing and virtualization led to servers running multiple virtual workloads which are commonly migrated from one server to another. Furthermore, storage replication and new applications that rely on multiple parallel workloads distributed across many servers, generate significant inter-server traffic [1,2]. The increasing volume of east-west traffic calls for lower latency interconnection fabrics in future data centers; traditional multi-tiered topologies with a high degree of oversubscription are now replaced by flatter full-bisection bandwidth architectures [3], such as the leaf-spine shown in Fig. 1. This topology enables scaling to high port counts while delivering full bisection bandwidth and it features low and predictable inter-rack latency as all paths are equidistant and also the shortest in length [4,5]. However, further bandwidth scaling is limited by the number of high-speed signal pins on electronic chips [6]. Optically switched networks exploiting wavelength-division multiplexing (WDM) enable increasing the transmission capacity by orders of magnitude effectively breaking the bisection bandwidth bottleneck.

Optical circuit switching has been proposed to increase the bisection bandwidth of the data center network and reduce its cost and complexity through prototypes such as c-Through [7], Helios [8], Mordia [9] and Proteus [10]. However, because the optical switches used are micro-electro-mechanical systems (MEMS), their reconfiguration time ranges from microseconds to tens of milliseconds. Hence, they are

* Corresponding author.
E-mail address: paris.andreades.09@ucl.ac.uk (P. Andreades).

https://doi.org/10.1016/j.osn.2018.11.005
Received 15 August 2018; Received in revised form 14 November 2018; Accepted 18 November 2018
Available online 21 November 2018

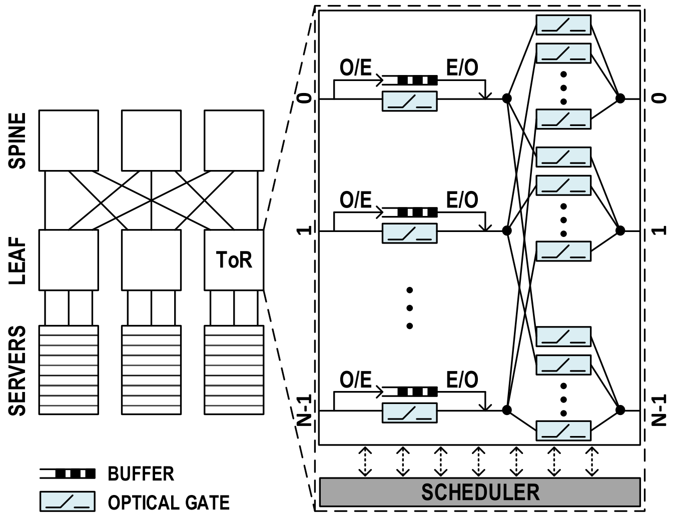

Fig. 1. Leaf-spine data center network and proposed optical switch architecture. In this example the optical switch is an input-buffered crossbar built using optical gate elements. Semiconductor optical amplifier (SOA) technology could be used to implement an optical gate.

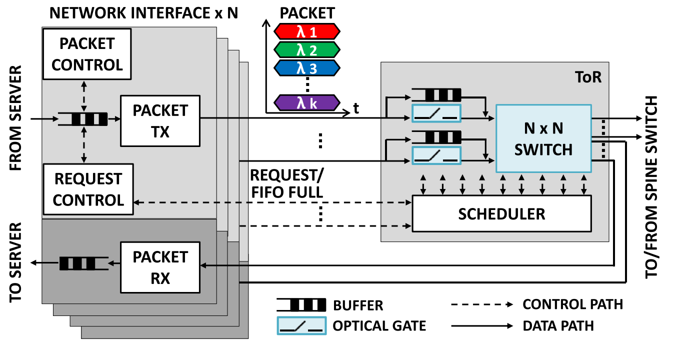

Fig. 2. The system concept. Server-side "send and forget" network interfaces and optical top-of-rack (ToR) switch with input electronic buffers to avoid packet loss. Every transmitted packet is divided into k wavelengths to further reduce latency. The ToR scheduler reconfigures the switch in nanoseconds.

best used when the traffic exhibits high stability [7–9] and thus are applied at the top layers of the network hierarchy. Control plane-wise, complex scheduling algorithms are used to estimate traffic demand to improve circuit utilization, incurring high latency in addition to arbitration delay and the round-trip and processing overheads due to switch path/circuit acknowledgement signaling.

Optical switches with nanosecond reconfiguration times have been demonstrated for packet switching. They can be built using micro-ring resonators [11,12], semiconductor optical amplifiers (SOAs) [13,14], Mach-Zehnder Interferometers (MZIs) [15] or by using a combination of technologies [16–19]. They can handle rapidly-changing bursty traffic as well and could be widely deployed in the network at different layers. Also packets are forwarded to the switch without waiting for acknowledgement signaling, effectively speculating on switch paths. The challenge, however, is building a control plane that can operate on packet timescales. The optical shared memory supercomputer interconnect system (OSMOSIS) demonstrator [20] and the optical cut-through (OpCut) switch [21] use centralized control and an iterative matching algorithm to offer high maximum throughput, at the cost of high scheduler complexity. On the other hand, switching systems such as the Data Vortex [22] and the scalable photonic integrated network (SPINet) [23] distribute control to the switch building blocks to simplify scheduling and improve scalability, at the expense of reduced global fairness and port throughput. Yet another approach is to build an optical network architecture which uses wavelength routing to avoid the need for global scheduling, for example [24,25]. This approach tends to increase optical component counts and, in particular, requires the use of costly wavelength conversion. In previous work [26] we experimentally demonstrated an FPGA-implemented centralized control plane for a 32-port SOA-based crossbar switch, enabling optical packet switching with a 75 ns end-to-end latency. At this size, the switch is suitable for top-of-rack (ToR) application at the leaf layer, as shown in Fig. 1, however scaling to larger sizes it could be used at the spine layer too. The proposed optical switch architecture uses electronic buffers at the input ports to reduce the minimum latency while avoiding packet loss.

The main focus of this paper is to present our high-speed control plane design. We implement it on FPGA boards, identify the end-to-end latency components including control plane contributions and discuss challenges on the minimum scheduling delay. We demonstrate experimentally nanosecond switching by means of high speed scheduling and speculative transmission of packets delivered across parallel WDM channels. We evaluate the minimum end-to-end latency and discuss the significance of control plane synchronization. Furthermore, we present a new highly-parallel output-port allocation circuit design for our switch scheduler and implement it on the same FPGA board as the experimental scheduler to quantify the improvement in scheduler delay and scalability. Finally, a cycle-accurate network emulator has been developed to perform an extensive latency analysis and comparison between the two scheduler allocation circuits. In this work, control plane demonstrations are for a 32×32 optical crossbar switch.

2. System concept

The system concept is shown in Fig. 2. It is based on a nanosecond-reconfigurable $N \times N$ optical switch with electronic buffers, implemented as first-in first-out (FIFO) queues, at the input ports. An electronic scheduler is co-located with the switch and it is responsible for the switch configuration. Implementing buffers at the switch inputs reduces the end-to-end latency and avoids packet loss, as discussed below.

Every new packet is first stored in a FIFO queue in the network interface at the server. When the queue is not empty, the request control reads the destination of the head-of-line (HOL) packet and issues an output port request to the scheduler. Next, once the guard time has expired, the packet control releases the packet speculatively; it does not wait for an output port grant from the scheduler. This eliminates the transport delay of a grant back to the server and also the overheads for grant synchronization and processing, effectively reducing the control overhead which in turn reduces the total latency [27]. Therefore, the network interfaces "send and forget" packets as opposed to storing them for possible re-transmission. The guard time ensures that packets will reach the switch as soon as it has been configured.

The transmitted packets are divided into segments each serialized onto a different wavelength using for example a dedicated silicon photonic transceiver [28], integrated on the server chip. The segments are multiplexed onto a WDM link and the entire wavelength-parallel structure is transported as one unit, as illustrated in Fig. 2. This process, known as wavelength striping, increases the input port capacity (number of wavelengths × wavelength bit rate) and reduces the packet (de)serialization latency (packet size ÷ port capacity), compared to related work [21,29] where packets are transmitted on a single serial channel. Also, for a given bit rate, the number of parallel wavelength channels bonded could be dynamically reconfigured to support variable-size packets at low serialization latencies.

In case of output port contention, the packet is stored at the switch buffer, instead of being dropped, and then recirculated to the output port in a subsequent switch configuration cycle. The scheduler always gives priority to packets buffered at the switch over new ones at the server network interfaces to maintain the packet ordering. Storing packets at the switch allows for shorter transport overheads of those packets to the switch output and the corresponding control signals between the buffers and the scheduler. The switch buffers are electronic and therefore optical to electrical (O/E) and electrical to optical (E/O) conversions are required. Although fiber delay line buffers could have been used at the switch inputs for all-optical switching, using fast dense electronic memory and emerging integrated silicon photonic WDM transceiver technology [28] is more compatible with the trend towards electronic and photonic integration. There is a one-to-one matching between the server network interfaces and switch recirculation buffers to simplify the scheduling algorithm and also the switch architecture, compared to [21,29] where packets at an input port may be stored in many different buffers.

A FIFO full signal is asserted when a switch buffer queue is almost full and the corresponding server network interface pauses transmission. This is the only control signal from the switch to a server and it is a single bit asserted when there is one free slot in the buffer, to account for a packet possibly in transit. This backpressure mechanism avoids buffer overflowing and packet loss.

In summary, ultra-low latency is achieved through a) high speed switch scheduling, b) speculative packet transmission (enabled by electronic buffering at the switch) and c) wavelength-striped packets.

3. Experimental demonstration

The purpose of the experiment is to present the high-speed control plane design which enables ultra-low latency optical switching for the system concept described above. We start by identifying the control plane latency components before describing the overall experimental setup.

3.1. Control plane design

The setup used for the control plane demonstration includes two FPGA boards and a 2 × 2 optical crossbar built out of discrete optical components, as shown in Fig. 3. FPGA#1 represents the server-side network interfaces and runs at a clock period T_{tx} while FPGA#2 represents the scheduler located at the switch, running at a clock period $T_{scheduler}$. The control plane extends across both FPGA boards; it includes the request and packet controllers in the network interfaces and the scheduler circuit. Its delay is therefore given by the sum of the individual control processes taking place in both the network interfaces and the scheduler:

1. Request generation: The request controller reads the destination of the HOL packet and generates a request. This takes one network interface clock cycle, T_{tx}.
2. Cable propagation delay: The request is transported to the scheduler out-of-band taking t_{cable}, given by the cable delay between server and switch. The request consists of only the destination output port and a valid bit and hence can be efficiently transmitted using parallel electronic connections, similar to how PCI-e networks are commonly used for control and management.
3. Request synchronization: In this demonstration, we assumed that the network interfaces and scheduler are asynchronous as is the case in today's commercial systems. The request synchronizer has a worst case delay of two $T_{scheduler}$ when the request arrives slightly after the scheduler clock edge. Synchronization latency is discussed further in the experimental results section.
4. Allocation: The optimum switch configuration given the existing switch paths and new requests is determined. In this demonstration allocation takes one $T_{scheduler}$ but forms the critical path in the scheduler design and is discussed in more detail below.
5. Switch configuration control: New grants made by the allocation circuit are added and expired grants are removed. The new grants are used to generate the switch configuration, including the read-enable signals for the switch buffers. Any requests not granted translate into write-enable signals for packet buffering at the switch. This takes one $T_{scheduler}$ plus the switch reconfiguration time, t_{switch}, which is the time needed to turn on a single SOA.

Transporting requests out-of-band in digital format eliminates the control overhead for in-band header/label detection. The label processor demonstrated in Ref. [30] incurs 100s of nanoseconds delay due to sampling frequency limitations imposed by analog-to-digital conversion, when implemented entirely on an FPGA board.

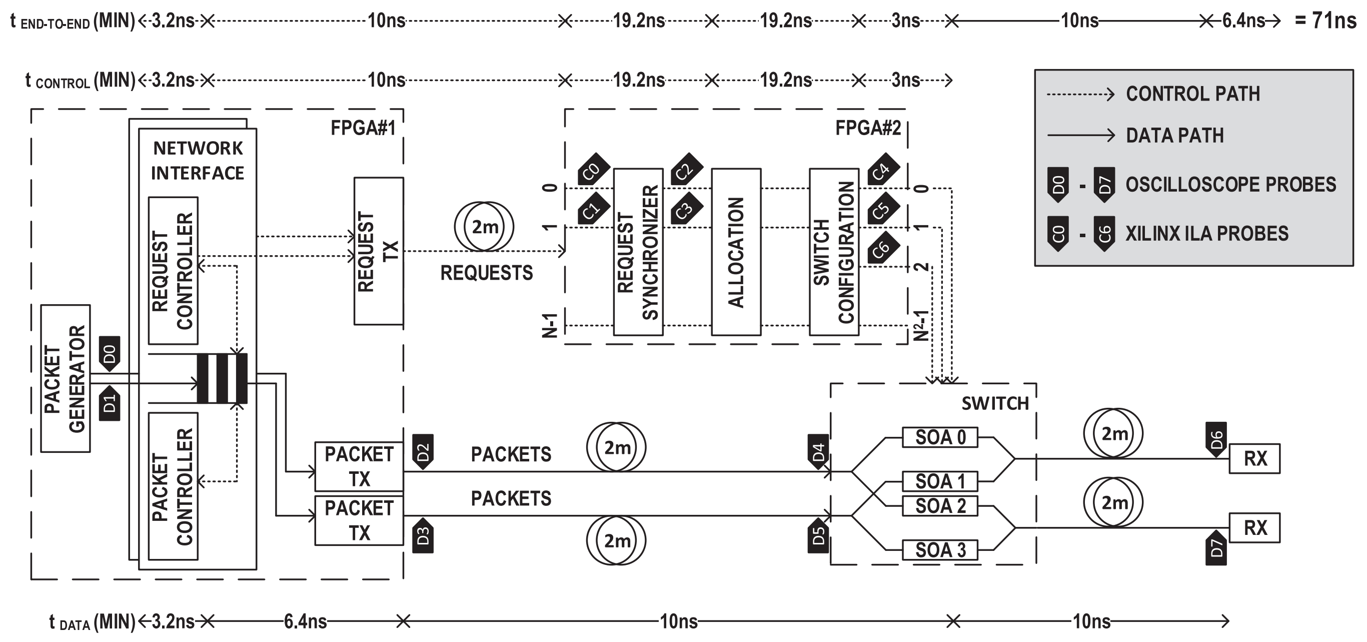

Fig. 3. The experimental setup. Control path and data path latency contributions and the minimum end-to-end latency are marked on the figure. Oscilloscope probes D0-D7 are placed along the data path. Xilinx Integrated Logic Analyzer (ILA) probes C0-C6 are implemented onto the scheduler FPGA board.

Having presented the components of the control plane design and quantified their respective delay contributions, it is understood that the total control latency is a function of the network interface and scheduler clock periods, T_{tx} and $T_{scheduler}$. The minimum control plane latency is deterministic and occurs for the minimum T_{tx} and $T_{scheduler}$ and under no contention when a packet is granted the requested switch output port:

$$t_{control_{(\min)}} = T_{tx_{(\min)}} + t_{cable} + 4T_{scheduler_{(\min)}} + t_{switch} \tag{1}$$

The minimum end-to-end latency, $t_{end-to-end_{(\min)}}$, is also deterministic. As shown in Fig. 3, it is the sum of $t_{control_{(\min)}}$ plus the fiber transport delay (t_{fiber}) from the switch to the receiver and the (de)serialization latency (t_{serial}) of the wavelength-striped packet at the receiver, which is effectively its head-to-tail latency:

$$t_{end-to-end_{(\min)}} = t_{control_{(\min)}} + t_{fiber} + t_{serial} \tag{2}$$

The minimum end-to-end latency does not include any packet buffering delay at the switch, since it occurs under no contention. It is dominated by $t_{control_{(\min)}}$; while the control plane processes a request, the corresponding packet in the data path is held by the packet controller at the network interface. When this guard time expires, the packet controller releases the packet for transmission so that it reaches the switch as soon as that has been configured.

The factor $4T_{scheduler_{(\min)}}$ is the scheduling delay; $2T_{scheduler_{(\min)}}$ for request synchronization (worst case) and $2T_{scheduler_{(\min)}}$ for allocation and switch configuration. It dominates $t_{control_{(\min)}}$ which in turn accounts for the majority of $t_{end-to-end_{(\min)}}$. The scheduler critical path is in the (output-port) *allocation circuit*, which means $T_{scheduler_{(\min)}}$ is determined by the minimum clock period of the allocation circuit. We achieve a low minimum clock period by having combinational logic blocks a) running in *parallel* and b) placed between registers dividing allocation in stages (*pipelining*).

Fig. 4 shows a simplified block diagram for the output-port allocation circuit used in the demonstration. Allocation is performed based on the well-known *round-robin* arbitration principle; a currently granted request gets the lowest priority in the next round of arbitration. The circuit has 2 pipeline stages, resulting in a 2-clock cycle allocation (including switch configuration). In the first pipeline stage two request matrices, one for new server requests and one for switch buffer requests, are read in and registered. Requests are initially arranged per input port and each is $\log_2(N)$-wide to address the destination output port. The size of each request matrix is therefore $N\log_2(N)$. In the second pipeline stage, output-port arbitration and new grants generation takes place, known as allocation. First, a request generator rotates each request matrix to arrange requests per output port, so that they can be read by the arbiters, and then multiplexes the two matrices into a single $N \times N$ matrix, R. It gives priority into the R matrix to requests from the switch buffers to maintain packet ordering and reduce packet

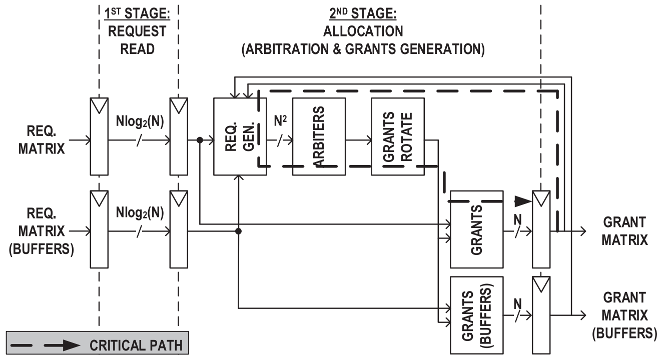

Fig. 4. Single-stage output-port allocation circuit block diagram.

delay. Also it removes any granted requests, using a feedback from the current grant matrix. Next, the R matrix is divided and fed to N N-bit round-robin arbiters to create the new $N \times N$ grant matrix G. Finally, G is de-rotated to rearrange grants per input port and then de-multiplexed into two output N-bit vectors, one for the new packets and one for the those buffered at the switch. The two grant vectors are registered and will be used next by the switch configuration controller.

In the allocation circuit, the feedback from the grant register to the request generator logic forms the critical path. It extends from the grant register, through one of the arbiters and back to the grant register itself, as marked in Fig. 4. Each N-bit arbiter is composed by N bit slices chained together [31]. It is implemented as a programmable priority encoder using carry-look-ahead to speed up propagation through the slice chain, as proposed by Ref. [32].

The entire scheduler, including the synchronization, allocation and switch configuration circuits, was designed in SystemVerilog and implemented onto the Xilinx Kintex-7 XC7K325T FPGA board, using the Xilinx Vivado tools. In this demonstration a 32×32 crossbar switch was assumed for which $T_{scheduler_{(\min)}} = 9.6$ ns.

The constraints on the minimum network interface clock period, $T_{tx_{(\min)}}$, are due to the bit rate per lane on the wavelength-striped data path and the width of the parallel bus driving the serializer. For this demonstration, 10 Gb/s per lane and a 32-bit parallel bus gave $T_{tx_{(\min)}} = 32\ bits/(10\ Gb/s) = 3.2$ ns. The network interface circuits were implemented onto the Xilinx Virtex-7 XC7VX690T FPGA board with this clock constraint.

For a 2-meter distance between the switch and the transmit and receive ends, emulating server to top-of-rack connections, $t_{cable} = 10$ ns may be assumed. Also, for the SOAs used in the demonstration, $t_{switch} = 3$ ns. Therefore, using equation (1) above, the minimum control plane latency is $t_{control_{(\min)}} = 54.6$ ns. The scheduler clock period dominates this value as $4T_{scheduler_{(\min)}} = 38.4$ ns or 70.3% of the total control plane latency. The penalty for asynchronous operation is $2T_{scheduler_{(\min)}} = 19.2$ ns or 35.2% of the total control plane latency.

For wavelength-striped 64-Byte (min. Ethernet size) packets in a WDM system using 8 wavelengths and each modulated/demodulated at 10 Gb/s, $t_{serial} = 6.4$ ns. Hence, assuming $t_{fiber} = 10$ ns, the expected $t_{end-to-end_{(\min)}}$ is 71.0 ns, according to equation (2).

3.2. Experimental setup

The experimental setup is shown in Fig. 3. In order to avoid clock distribution, the two FPGA boards are asynchronous, running from independent crystal oscillators. As discussed in the previous section, the control plane logic on FPGA#1 and FPGA#2 ran at clock periods of $T_{tx_{(\min)}} = 3.2$ ns and $T_{scheduler_{(\min)}} = 9.6$ ns respectively. The request structure consisted of a 5-bit destination field and a valid bit and parallel electronic connections were used to transfer requests between the two boards. Since the control plane latency is the main focus of this work, the data plane was simplified in the following ways:

1. Although the scheduler was implemented for a 32×32 crossbar switch, only a 2×2 switch was implemented in the setup.
2. Since the minimum end-to-end latency occurs when a packet experiences no contention, no recirculation buffers were connected to the switch inputs. However, the scheduler still generates the buffer control signals and input SOA control signals.
3. Although in practice data would be striped onto multiple wavelengths, only a single 10 Gb/s enhanced small form-factor pluggable (SFP+) optical transceiver per port was used and only 64 bits per packet were transmitted, effectively emulating a packet serialization latency of 6.4 ns. This is equivalent to transmitting 64 B packets at 80 Gb/s.

Two network interfaces together with a packet generator and two optical transceivers (SFP+) were implemented onto FPGA#1 (Xilinx

XC7VX690T). The generator periodically constructs 64-Byte packets including a valid-bit and source, destination and payload fields. We chose this packet size to represent minimum Ethernet packets for which the minimum end-to-end latency occurs. In practice, variable-size packets may be transmitted and the number of wavelengths may be changed dynamically to maintain a low t_{serial}. A linear feedback shift register is used to assign a pseudo-random payload to the packets. Because the two FPGA boards are asynchronous, the request controller extends the duration of requests by a configurable number of clock cycles so that the slower scheduler can detect them. In this experiment the duration of each request was set to $4T_{tx_{(\min)}}$. Also configurable is the guard time during which the packet controller holds onto a packet. It was set to $10T_{tx_{(\min)}}$ to account for $t_{control_{(\min)}}$ and additional fiber delays in the switch itself because it was implemented with off-the-shelf discrete components (splitters/combiners, SOAs).

The scheduler circuit was implemented onto FPGA#2 (Xilinx XC7K325T). The width of the switch configuration signals was set to $2T_{scheduler_{(\min)}}$ to accommodate both the packet length and the t_{switch} overhead, while also accounting for the uncertainty in the request arrival at FPGA#2 with respect to the scheduler clock. This is discussed in more detail in the next section. A Xilinx Integrated Logic Analyzer (ILA) core was also implemented onto FPGA#2 for real-time monitoring of the scheduler signals, using probes C0-C6 as marked in Fig. 3.

The packet timings are captured on a high-speed oscilloscope at several points along the data paths by placing probes D0-D7 as indicated in Fig. 3 and using optical receivers where necessary.

4. Experimental results

As depicted in Fig. 5, the control plane demonstration consists of two scenarios: a) switching packets from a single network interface to either of the two output ports of the switch and b) switching packets from both network interfaces to a single output port.

4.1. Scenario A

The packet generator injects packets periodically, once every 48 ns, with their destination alternating between the two possible output ports of the switch. The purpose of this exercise is to verify successfully packet switching and also measure the minimum end-to-end latency.

4.1.1. Control path signals

Fig. 6 shows the timing of the control processes at the scheduler, captured using ILA probes as illustrated in Fig. 3. First, the input requests (probe C0) are successfully detected (only valid bits are

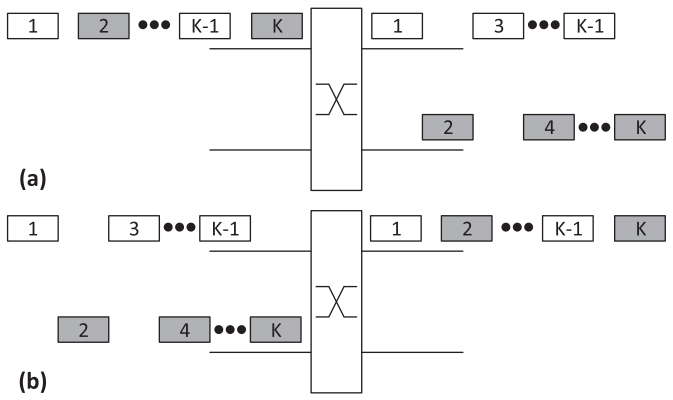

Fig. 5. Experimental scenarios: a) switching packets from an input port to two output ports and b) switching packets from two input ports to an output port.

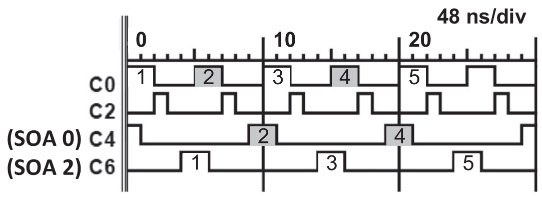

Fig. 6. Scheduler shown granting the same network interface to two switch output ports in turn, one at a time. Traces are labeled according to the probe placement shown in Fig. 3. Each subdivision is 1 $T_{scheduler_{(\min)}}$.

shown). It then takes two clock cycles to go through the synchronization stage before they are detected by the allocation circuit (probe C2). Finally, it takes two more clock cycles to complete allocation and produce the switch configuration signals for the two output SOAs (probes C4 and C6). The width of the configuration pulses is two clock cycles. As shown in the figure, only one output SOA is active at a time and hence packets are switched to the corresponding output port. The packets appear at the same output port on every other scheduling round, as expected.

4.1.2. Data path signals

Fig. 7 shows the packet timings at different points along the data paths from the server interface to the two output ports, as captured on the oscilloscope. The falling edge of a packet's valid bit entering the network interface marks the starting point for the end-to-end latency measurement. It takes 36 ns for the packets to leave the FPGA and 12 ns more to arrive at the switch SOA. It then takes 28 ns to reach the receivers and 6.4 ns more to deserialize the packet at the receiver. This brings the end-to-end latency to 82.4 ns compared with the expected value of 71.0 ns. This is due to additional PCB tracks in the control path and fiber pigtails in discrete components in the data path, such as the 2:1 combiners at the switch output ports and off-the-

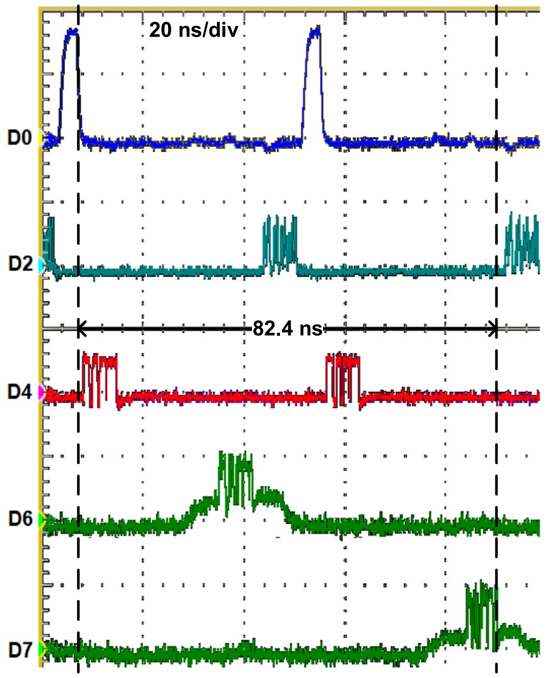

Fig. 7. Data paths from a network interface to the 2 output ports of the switch. Traces are labeled according to the probe placement shown in Fig. 3. The minimum end-to-end latency measurement is marked on the figure.

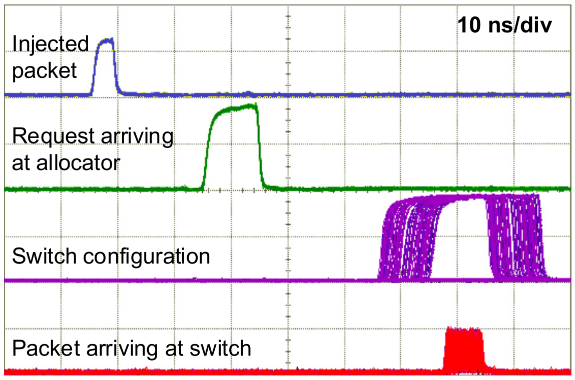

Fig. 8. Request arrival uncertainty due to asynchronous operation and its effect on the switch configuration signals, captured on the oscilloscope using 5-second persistence.

shelf optical receivers, which increase t_{cable} and t_{fiber} to more than 10 ns each. The amplification effect of the SOAs is exhibited as a voltage offset every time an SOA is switched ON. Although this could have been removed with an optical filter, none was applied in order to show the packet arrival relative to the scheduler SOA configuration pulses.

The control plane is asynchronous, since the two FPGA boards run from different clock sources, thus it is uncertain when exactly a request arrives at the scheduler board with respect to the scheduler clock edge. The offset can lie anywhere from 0 ns up to a maximum of 1 $T_{scheduler}$, which is the case when the request falls just after the clock rising edge. Hence the total delay for request synchronization can be up to $2T_{scheduler}$ in the worst case. The $10T_{tx_{(\min)}} = 32$ ns guard time accounts for the worst case request synchronization delay. The effect of the uncertainty in the request arrival time on the scheduler output configuration signals was captured on the oscilloscope using a 5-second persistence, as shown in Fig. 8. The configuration pulses need to be broadened by at least one extra clock cycle depending on the length of $T_{scheduler_{(\min)}}$. In this case, $T_{scheduler_{(\min)}} = 9.6$ ns is long enough to accommodate both the packet duration and the SOA switch-ON time (t_{switch}). As a result, a window of $3T_{scheduler_{(\min)}} = 28.8$ ns is defined during which a switch configuration may occur. The packet is shown here arriving at the switch/SOA at a time well aligned with the configuration pulse, indicating correct guard time calculation.

4.2. Scenario B

In this scenario all packets generated are destined to the same output port but fed to the two network interfaces with a constant time difference. Staggering the packets enables investigating the minimum time spacing between packets of different network interfaces, in order to measure the switch throughput.

4.2.1. Control path signals

Requests from the two network interfaces (probes C0 and C1) are shown in Fig. 9 arriving at the scheduler with a fixed time difference, reflecting the packet staggering applied. Because the requests generated at each network interface are 12.8 ns wide, some of them are registered twice depending on when they arrive relative to the scheduler clock edge. The scheduler is shown here granting the output port (probes C4 and C5) to the network interface that requested it first and then, in the next scheduling round, granting to the other one. The requests from the two network interfaces can be as close as $10T_{tx_{(\min)}} = 32$ ns to one another. This is due to the window of 28.8 ns over which a switch configuration could occur due to the asynchronous operation, as discussed above. On the ILA, which runs at the scheduler clock period, the minimum request spacing appears as $3T_{scheduler_{(\min)}} = 28.8$ ns.

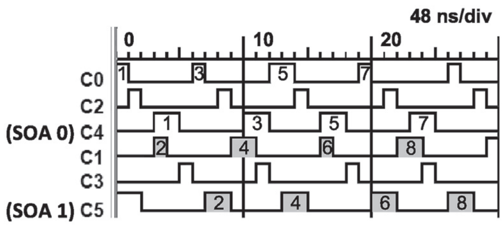

Fig. 9. Scheduler shown granting the same switch output port to two network interfaces in turn, one at a time. Traces are labeled according to the probe placement shown in Fig. 3. Each subdivision is 1 $T_{scheduler_{(\min)}}$.

4.2.2. Data path signals

Fig. 10 shows the minimum time spacing between packets from different network interfaces. That is 32 ns and it is mainly due to the control plane being asynchronous. As a result, in this particular demonstration setup, the throughput is $(6.4/32) \times 100 = 20\%$. With a synchronous control plane, packets from different sources could be

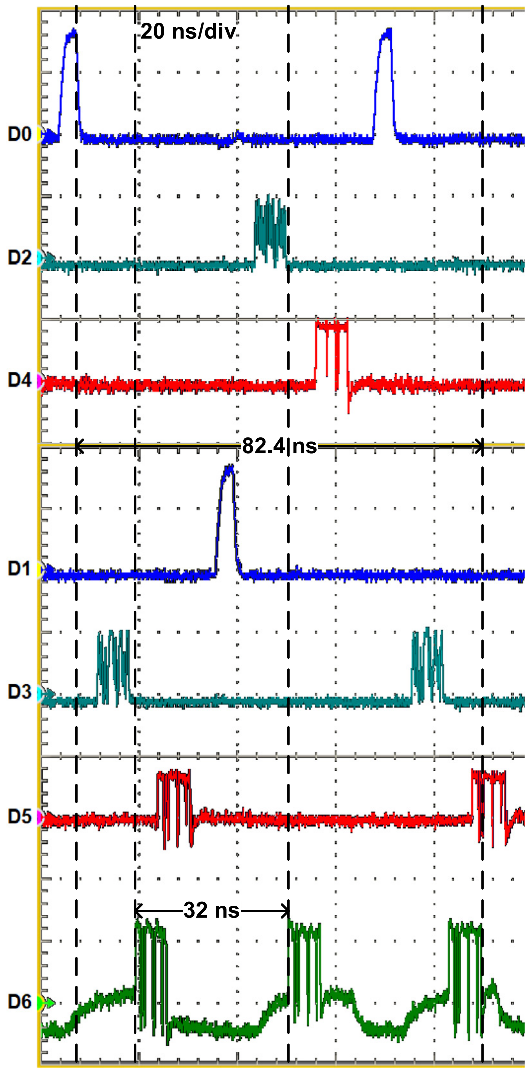

Fig. 10. Data paths from the 2 network interfaces to an output port of the switch. Traces are labeled according to the probe placement shown in Fig. 3. The minimum end-to-end latency and the minimum packet separation measurements are marked on the figure.

spaced by $T_{scheduler}$, which would bring the maximum throughput to $(T_{scheduler} - t_{switch})/T_{scheduler} \times 100 = 69\%$. Nonetheless, the focus here is to demonstrate the latency rather than the throughput of our switch system concept and the implementation of a synchronous control plane is outside the scope of this work. The delay characteristics along each data path are the same as in the previous section. The packets coming from the two different inputs are all successfully received at the output port.

4.3. Asynchronous control plane considerations

Asynchronous control plane communication increases the end-to-end latency because a request synchronization stage needs to be included in the scheduler circuit. In our experimental demonstration, this stage contributes 27% of the total latency. It also increases the switch configuration duration (to two scheduler clock cycles in the case of this demonstration for 64 B packets) to take into account the uncertainty in the request arrival time at the scheduler. This acts to decrease the switch throughput as no new packets may traverse the switch during that time. Synchronous operation would significantly reduce latency at the expense of additional wiring to distribute a clock signal. In practice, synchronization would also be necessary for data recovery on the receiving end, which could incur a significant latency penalty if implemented using conventional clock recovery circuits on a per packet basis. While full frequency and phase synchronization at the bit level has not been demonstrated on the scale of data centers, techniques such as injection clock recovery, either electronic [33,34] or optical [35], and wavelength-striped source-synchronous operation [36,37] are potential low-latency solutions to the receiver synchronization problem. These techniques still involve delay due to clock domain crossings. However, the proposed leaf-spine network with optical leaf switches, avoids any clock domain crossings at this level of the hierarchy.

5. Two-stage allocation circuit

The minimum allocation circuit clock period is crucial on the achievable lower bound of the end-to-end latency in our switch system concept. In the scheduler used in the experimental demonstration, although the allocation circuit (Fig. 4) is a two-stage pipeline, it performs the functions of arbitration and new grants generation in a single pipeline stage, thus resulting in a long critical path. In Ref. [38] we briefly introduced a faster allocation circuit, shown in Fig. 11. Here we discuss in detail the circuit implementation, evaluate reductions in control and end-to-end latencies and investigate scheduling scalability compared to single-stage allocation.

The new allocation circuit design reduces the critical path in the following ways:

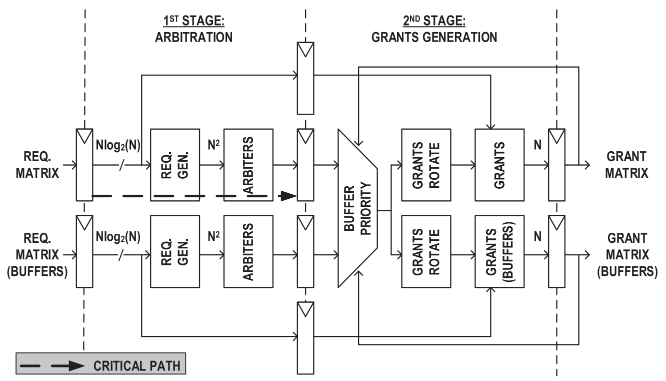

Fig. 11. Two-stage output-port allocation circuit block diagram.

1. Arbitration and new grants generation are divided in the two pipeline stages.
2. Arbitration is performed separately and in parallel for new server requests and for requests coming from the switch buffers. This enables removing the logic for multiplexing and prioritizing requests when generating the requests for the arbiters. The priority logic is moved to the 2nd pipeline stage.
3. Feedback from the current grant matrix is moved to the 2nd pipeline stage and applies to the logic for multiplexing and prioritizing the new grants. This means that instead of dropping already resolved requests, new grants are filtered out.

As a result, the critical path falls in the arbitration stage, from the input request register through one of the arbiters to the arbiter output register, as illustrated in Fig. 11. Each of the two input $Nlog_2(N)$ request matrices is fed to a request generator logic for translation into an arbiter readable format; the request matrix is expanded to $N \times N$ and then rotated so that requests are grouped per output port. The resulting request matrices R and R' are each divided and fed to N N-bit arbiters. Arbitration is performed in parallel for R and R'. The two output grant matrices, G and G', are registered and in the next pipeline stage are multiplexed into a single $N \times N$ matrix, giving priority to requests from the switch buffers. Feedback from the current grant matrices is used to filtered out any grants already issued. Finally parallel logic blocks are used to de-rotate the multiplexed matrix, arranging the grants per input port, and then produce N-bit grant vectors for the server requests and requests from the switch buffers.

The scheduler for both allocation circuit designs is implemented on the Xilinx Kintex-7 XC7K325T FPGA board for different $N \times N$ switch sizes and the timing results are shown in Fig. 12. In both designs, the arbiter module is in the critical path and is composed as a cascade of N arbiter bit slices. Therefore, $T_{scheduler_{(\min)}}$ increases with the switch size. The single-stage allocation circuit hinders scheduling scalability beyond a switch size of 32×32. However, for the same switch size, the two-stage allocation circuit reduces $T_{scheduler_{(\min)}}$ by 42.7% giving $T_{scheduler_{(\min)}} = 5.5$ ns. At this clock period, the two-stage allocation circuit quadruples the switch size compared to the single-stage allocation circuit. More importantly, for $T_{scheduler} = 9.6$ ns it enables doubling the switch size to 64×64.

For our control plane experimental demonstration setup shown in Fig. 3, using the new two-stage allocation circuit and running it at $T_{scheduler_{(\min)}} = 5.5$ ns would bring the expected $t_{control_{(\min)}}$ and $t_{end-to-end_{(\min)}}$ down to 38.2 ns and 54.6 ns respectively, according to equations (1) and (2). Compared to using single-stage allocation, the reduction in the expected $t_{end-to-end_{(\min)}}$ is 23.1%.

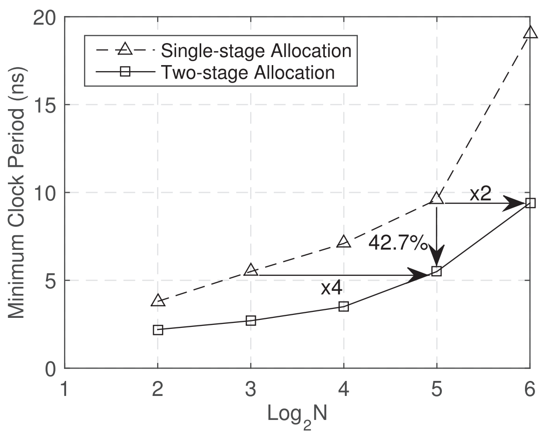

Fig. 12. Timing results for scheduler implementation onto the Xilinx Kintex-7 325T FPGA board using either of the two allocation circuits.

Table 1
Network model parameters.

Parameter	Value
Switch size (N)	32
T_{tx}	3.2 ns
$T_{scheduler}$	9.6 ns, 5.5 ns
t_{serial}	6.4 ns
t_{cable}	10 ns
t_{fiber}	10 ns
t_{switch}	3 ns
Request synchronization delay	19.2 ns, 11 ns
Configuration pulse width	19.2 ns, 16.5 ns
Measurement time	10 μs

Implementing our work on CMOS application-specific integrated circuit (ASIC) technology would reduce latency even further, for either allocation circuit being used in the scheduler. For comparison, an electronic 10 Gb/s 48-port cut-through packet switch has a minimum latency of 170 ns [39]. Including the same cable and packet serialization delays as in our experimental setup, its minimum end-to-end latency comes to 196.4 ns, without taking into account any control overheads on the transmit side.

6. Network emulation

In this section we do a further investigation of the end-to-end latency in our switch system concept, based on a network model developed in SystemVerilog. The latency performance is evaluated for both output-port allocation circuits presented above.

A full-size rack-scale network is emulated for a 32 × 32 optical top-of-rack (leaf) switch, including 32 network interfaces and 32 switch buffers. The depth of each buffer is set to 1024 packet entries to avoid packet loss at 100% load. We assume a 2 m fiber length from the rack to the switch and another 2 m fiber length from the switch back to the same rack or to the spine switch. We also assume wavelength-striped packets with t_{serial} = 6.4 ns. This means that for a fixed bit rate, for example 400 Gb/s, the number of wavelengths used may be changed dynamically according to the packet size to preserve t_{serial}. The model captures the latency characteristics of our control plane and adds the total fiber transport delay and packet deserialization delay at the receiver to calculate the end-to-end latency in the data plane. It includes the buffering latency contributions to both the control plane and data plane.

The latency performance is measured under a synthetic workload. A set of 32 independent packet sources are instantiated each generating packets randomly following a uniform inter-arrival time distribution. A universal load parameter is used to set the probability of a packet being generated in a clock cycle, thus controlling the capacity percentage the sources run at. A 100% load represents packets generated on every clock cycle, i.e. with no time gaps. The packet output port destinations are uniform random. Each packet source feeds a dedicated network interface module. Table 1 lists the parameter values used in the network emulation.

Fig. 13, shows how the average end-to-end latency varies as a function of the load for either allocation circuit being used. The two-stage allocation design performs better across all port loads. At low loads, both curves converge to their expected minimum end-to-end latencies as given by equation (2). As the load is increased the probability of contention gets higher and more packets are buffered at the switch, increasing the average end-to-end latency. At high loads, beyond 50%, the latency rapidly increases due to a greater number of packets experiencing head-of-line (HOL) blocking. With either allocation circuit being used the average latency at 100% load is 10 μs or less, under uniform random traffic. Nonetheless, the focus of this work is ultra-low latency switching at low loads which is verified by these results. Speculative

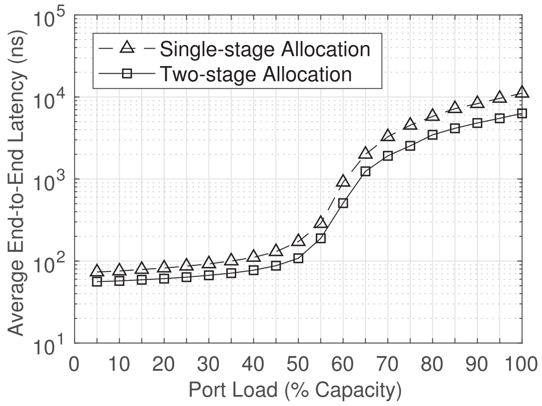

Fig. 13. Average end-to-end latency vs. port load under uniform random traffic. Network model assumes a 32 × 32 optical top-of-rack switch and 2 m fiber connections to the input and output ports. Packets are wavelength-striped with a 6.4 ns duration.

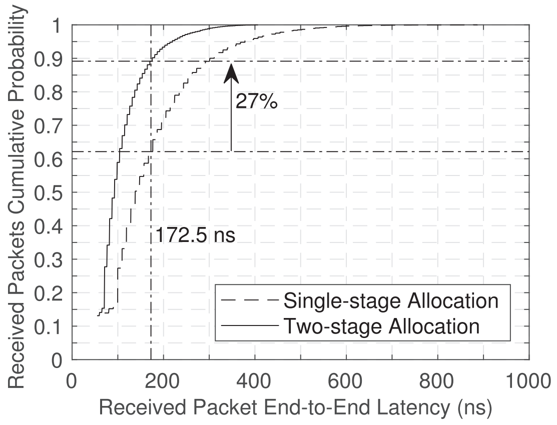

Fig. 14. Cumulative distribution of the received packet end-to-end latency, at 50% load under uniform random traffic. Network model assumes a 32 × 32 optical top-of-rack switch and 2 m fiber connections to the input and output ports. Packets are wavelength-striped with a 6.4 ns duration.

transmission is one of the main enablers in achieving this at the cost of increased buffering at high loads. The switch buffers are assumed to be deep enough (1024 packet entries in this emulation) to avoid packet loss at all loads.

The average latency at high loads could be reduced by changing the buffer placement at the switch; at every input port one virtual queue per output port may be implemented to store packets per destination, significantly reducing the HOL blocking and thus the buffering time and average latency [40]. The depth requirement of each queue would also be much smaller. Implementing virtual output queues (VOQs) however not only increases the complexity of the switch but also that of the scheduling algorithm, incurring additional delay. For instance, for VOQ switches, FPGA-implementations of parallel iterative round-robin-based (iSLIP) or weighted (ipLQF) arbitration have been recently reported [41] with hundreds of nanoseconds delay for scheduling alone and that for only a 16-port switch.

In Fig. 14 the cumulative distributions of the packet end-to-end latency for the two allocation circuits at 50% load are shown. Over the measurement window of 10 μs, more than 10% of the packets were received with the minimum end-to-end latency, for both allocation circuits. According to Fig. 13, the average end-to-end latency for the single-stage allocation circuit is 172.5 ns. At this value, the two-stage allocation circuit enabled switching 27% more packets. Furthermore, 95% of the packets were received with a latency in the range 71.0 ns–378.2 ns for the single-stage allocation circuit design against 54.6 ns–219.6 ns for the two-stage allocation design. Thus, for 95% of the packets, the longest possible latency was reduced by 158.6 ns. Moreover, two-stage allocation results in a shorter distribution tail; from 95% to 100% of the received packets, the increase in the longest possible end-to-end latency is 85.2% compared to 134.5% for single-stage allocation.

7. Conclusion

We have presented the implementation details of our low-latency control plane and demonstrated it experimentally for a 32×32 optical crossbar switch. For this switch size, in a data center top-of-rack application, the switch scheduling accounts for 70.3% of the minimum control plane delay which in turn accounts for 76.9% of the minimum end-to-end head-to-tail latency, for wavelength-striped 6.4 ns long packets. The output-port allocation circuit determines the minimum scheduling delay and therefore we presented here a new circuit design which divides allocation into two stages to reduce the critical path in the design. As a result the minimum clock period is reduced from 9.6 ns to 5.5 ns, decreasing the scheduling contribution to the minimum control plane latency to 57.6%. Apart from fast scheduling, the control plane delay is also reduced using speculative transmission and storing packets at the switch inputs. By speculating on output-port grants the control path is reduced as otherwise grant synchronization and processing would be required at the transmit side. Buffering packets at the switch keeps them close to the scheduler and hence reduces the transport overhead of requests and grants between the scheduler and the buffers at the switch.

Overall, in our FPGA-based control plane demonstration a minimum end-to-end latency of 82.4 ns was achieved, of which 11.4 ns was due to additional fiber/cable delays. For 2-meter fiber connections to the switch ports, the minimum latency is 71.0 ns or 54.6 ns when two-stage allocation is used and further reduction is expected when our work is implemented on CMOS ASIC technologies. For comparison, equivalent state-of-the-art electronic switch systems have a minimum latency around 200 ns. More importantly, the two-stage allocation circuit improves scheduler scalability. Indeed, for the same clock period it enables scaling from a 32×32 to a 64×64 switch size, considerably increasing the number of hosts in the leaf-spine data center network.

A rack-scale 32×32 network emulation was used to measure the packet end-to-end latency in our optical switch system concept, under uniform random traffic. The scheduler based on two-stage allocation showed a lower average latency at all port loads. Moreover, for the average latency at 50% port capacity, it allowed switching 27% more packets, over a measurement period of 10 μs.

Synchronous control plane operation would improve both the latency and throughput in our optical switch system, at the cost of extra wiring to distribute the clock signal. Practical systems also require receiver synchronization for data recovery, but techniques such as injection clock recovery and wavelength-striped source-synchronous operation could be used to keep the latency penalty small.

Acknowledgement

The authors would like to thank Yujia Wang, Shiyun Liu and Jeffrey Shen for their contributions towards designing and implementing the control plane. The work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) TRANSNET program grant EP/R035342/1 and the Fellowship grant to Philip Watts. It was also partly supported by the EU H2020 project dRedBox under grant agreement no. 687632. The FPGA evaluation boards used in our experimental demonstration were provided by Xilinx.

References

[1] Cisco Systems, Inc, Cisco Global Cloud Index: Forecast and Methodology, 2015 - 2020, White Paper, Cisco Systems, Inc., 2016.

[2] G. Lee, Cloud Networking: Understanding Cloud-based Data Center Networks, Morgan Kaufmann, 2014.

[3] N. Chrysos, F. Neeser, M. Gusat, C. Minkenberg, W. Denzel, C. Basso, M. Rudquist, K.M. Valk, B. Vanderpool, Large switches or blocking multi-stage networks? An evaluation of routing strategies for datacenter fabrics, Comput. Network. 91 (2015) 316–328. https://doi.org/10.1016/j.comnet.2015.08.029.

[4] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, Comput. Commun. Rev. 38 (4) (2008) 63–74, https://doi.org/10.1145/1402946.1402967.

[5] M. Alizadeh, T. Edsall, On the data path performance of leaf-spine datacenter fabrics, in: IEEE Hot Interconnects (HOTI), 2013, pp. 71–74, https://doi.org/10.1109/HOTI.2013.23.

[6] N. Zilberman, P.M. Watts, C. Rotsos, A.W. Moore, Reconfigurable network systems and software-defined networking, Proc. IEEE 103 (7) (2015) 1102–1124, https://doi.org/10.1109/JPROC.2015.2435732.

[7] G. Wang, D.G. Andersen, M. Kaminsky, K. Papagiannaki, T.E. Ng, M. Kozuch, M. Ryan, c-Through: Part-time optics in data centers, in: Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM '10, ACM, New York, NY, USA, 2010, pp. 327–338, https://doi.org/10.1145/1851182.1851222.

[8] N. Farrington, G. Porter, S. Radhakrishnan, H.H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, A. Vahdat, Helios: a hybrid electrical/optical switch architecture for modular data centers, in: Proceedings of the ACM SIGCOMM 2010 Conference, SIGCOMM '10, ACM, New York, NY, USA, 2010, pp. 339–350, https://doi.org/10.1145/1851182.1851223.

[9] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, A. Vahdat, Integrating microsecond circuit switching into the data center, in: Proceedings of the ACM SIGCOMM 2013 Conference, SIGCOMM '13, ACM, New York, NY, USA, 2013, pp. 447–458, https://doi.org/10.1145/2486001.2486007.

[10] A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, Proteus: a topology malleable data center network, in: ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets), HotNets-IX, ACM, New York, NY, USA, 2010, pp. 1–6, https://doi.org/10.1145/1868447.1868455.

[11] A. Biberman, G. Hendry, J. Chan, H. Wang, K. Bergman, K. Preston, N. Sherwood-Droz, J.S. Levy, M. Lipson, CMOS-compatible scalable photonic switch architecture using 3D-integrated deposited silicon materials for high-performance data center networks, in: Optical Fiber Communication Conference and Exposition (OFC/NFOEC), 2011 and the National Fiber Optic Engineers Conference, 2011, https://doi.org/10.1364/OFC.2011.OMM2.

[12] A.W. Poon, L. Xianshu, X. Fang, C. Hui, Cascaded microresonator-based matrix switch for silicon on-chip optical interconnection, Proc. IEEE 97 (7) (2009) 1216–1238, https://doi.org/10.1109/JPROC.2009.2014884.

[13] I.H. White, E.T. Aw, K.A. Williams, H. Wang, A. Wonfor, R.V. Penty, Scalable optical switches for computing applications, [Invited], J. Opt. Netw. 8 (2) (2009) 215–224, https://doi.org/10.1364/JON.8.000215.

[14] O. Liboiron-Ladouceur, B.A. Small, K. Bergman, Physical layer scalability of WDM optical packet interconnection networks, J. Lightwave Technol. 24 (1) (2006) 262–270, https://doi.org/10.1109/JLT.2005.859852.

[15] 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, Y.A. Vlasov, Monolithic silicon integration of scaled photonic switch fabrics, CMOS logic, and device driver circuits, J. Lightwave Technol. 32 (4) (2014) 743–751, https://doi.org/10.1109/JLT.2013.2280400.

[16] Q. Cheng, A. Wonfor, R.V. Penty, I.H. White, Scalable, low-energy hybrid photonic space switch, J. Lightwave Technol. 31 (18) (2013) 3077–3084, https://doi.org/10.1109/JLT.2013.2278708.

[17] Q. Cheng, A. Wonfor, J. Wei, R.V. Penty, I.H. White, Modular hybrid dilated Mach-Zehnder switch with integrated SOAs for large port count switches, in: Optical Fiber Communications Conference and Exhibition (OFC), 2014, https://doi.org/10.1364/OFC.2014.W4C.6.

[18] N. Calabretta, W. Miao, K. Mekonnen, K. Prifti, K. Williams, Monolithically integrated WDM cross-connect switch for high-performance optical data center networks, in: Optical Fiber Communications Conference and Exhibition (OFC), 2017, https://doi.org/10.1364/OFC.2017.Tu3F.1.

[19] W. Miao, H. d. Waardt, R. v. d. Linden, N. Calabretta, Assessment of scalable and fast 1310-nm optical switch for high-capacity data center networks, IEEE Photon. Technol. Lett. 29 (1) (2017) 98–101, https://doi.org/10.1109/LPT.2016.2629087.

[20] R. Luijten, C. Minkenberg, R. Hemenway, M. Sauer, R. Grzybowski, Viable opto-electronic HPC interconnect fabrics, in: ACM/IEEE Supercomputing Conference, 2005, https://doi.org/10.1109/SC.2005.78.

[21] L. Liu, Z. Zhang, Y. Yang, Packet scheduling in a low-latency optical interconnect with electronic buffers, J. Lightwave Technol. 30 (12) (2012) 1869–1881, https://doi.org/10.1109/JLT.2012.2190971.

[22] Q. Yang, K. Bergman, Performances of the Data Vortex switch architecture under nonuniform and bursty traffic, J. Lightwave Technol. 20 (8) (2002) 1242–1247, https://doi.org/10.1109/JLT.2002.800330.

[23] A. Shacham, K. Bergman, Building ultralow-latency interconnection networks using photonic integration, IEEE Micro 27 (4) (2007) 6–20, https://doi.org/10.1109/MM.2007.64.

[24] J. Luo, S. Di Lucente, J. Ramirez, H.J.S. Dorren, N. Calabretta, Low latency and large port count optical packet switch with highly distributed control, in: Optical Fiber Communication Conference and Exposition (OFC/NFOEC) and the National Fiber Optic Engineers Conference, 2012, https://doi.org/10.1364/OFC.2012.OW3J.2.

[25] P. Grani, R. Proietti, S. Cheung, S.J.B. Yoo, Flat-topology high-throughput compute node with AWGR-based optical-interconnects, J. Lightwave Technol. 34 (12) (2016) 2959–2968, https://doi.org/10.1109/JLT.2015.2510656.

[26] P. Andreades, Y. Wang, J. Shen, S. Liu, 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), 2015, https://doi.org/10.1364/OFC.2015.W3D.5.

[27] P.M. Watts, S.W. Moore, A.W. Moore, Energy implications of photonic networks with speculative transmission, J. Opt. Commun. Netw. 4 (6) (2012) 503–513, https://doi.org/10.1364/JOCN.4.000503.

[28] 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, A.V. Krishnamoorthy, Ultra-efficient 10Gb/s hybrid integrated silicon photonic transmitter and receiver, Optic Express 19 (6) (2011) 5172–5186, https://doi.org/10.1364/OE.19.005172.

[29] X. Ye, Y. Yin, S. Yoo, P. Mejia, R. Proietti, V. Akella, DOS - a scalable optical switch for datacenters, in: ACM/IEEE Architectures for Networking and Communications Systems (ANCS), 2010, pp. 1–12.

[30] N. Calabretta, FPGA-based label processor for low latency and large port count optical packet switches, J. Lightwave Technol. 30 (19) (2012) 3173–3181, https://doi.org/10.1109/JLT.2012.2215840.

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

[32] P. Gupta, N. McKeown, Designing and implementing a fast crossbar scheduler, IEEE Micro 19 (1) (1999) 20–28, https://doi.org/10.1109/40.748793.

[33] B. Li, L.S. Tamil, D. Wolfe, J. Plessa, 10 Gbps burst-mode optical receiver based on active phase injection and dynamic threshold level setting, Commun. Lett. IEEE 10 (10) (2006) 722–724, https://doi.org/10.1109/LCOMM.2006.060099.

[34] J. Lee, M. Liu, A 20-Gb/s burst-mode clock and data recovery circuit using injection-locking technique, J. Solid-State Circ. 43 (3) (2008) 619–630, https://doi.org/10.1109/JSSC.2007.916598.

[35] J. Luo, J. Parra-Cetina, P. Landais, H.J. Dorren, N. Calabretta, Performance assessment of 40 Gb/s burst optical clock recovery based on quantum dash laser, Photon. Technol. Lett. IEEE 25 (22) (2013) 2221–2224, https://doi.org/10.1109/LPT.2013.2284529.

[36] C.E. Gray, O. Liboiron-Ladouceur, D.C. Keezer, K. Bergman, Multi-gigahertz source synchronous testing of an optical packet switching network, in: International Mixed-signals Test Workshop, Edinburgh, Scotland, 2006.

[37] M. De Wilde, O. Rits, R. Baets, J.V. Campenhout, Synchronous parallel optical I/O on CMOS: a case study of the uniformity issue, J. Lightwave Technol. 26 (2) (2008) 257–275, https://doi.org/10.1109/JLT.2007.909849.

[38] P. Andreades, P.M. Watts, Low latency parallel schedulers for photonic integrated optical switch architectures in data centre networks, in: European Conference on Optical Communication (ECOC), 2017, https://doi.org/10.1109/ECOC.2017.8345961.

[39] Cisco Systems, Inc, Cisco Nexus 3548 Switch Performance Validation, White Paper, Cisco Systems, Inc., December, 2012.

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

[41] I. Cerutti, J.A. Corvera, S.M. Dumlao, R. Reyes, P. Castoldi, N. Andriolli, Simulation and fpga-based implementation of iterative parallel schedulers for optical interconnection networks, IEEE/OSA J. Optic. Commun. Network. 9 (4) (2017) C76–C87, https://doi.org/10.1364/JOCN.9.000C76.