PULSE: Optical Circuit Switched Data Center Architecture Operating at Nanosecond Timescales

Joshua L. Benjamin, *Student Member, IEEE*, Thomas Gerard, *Student Member, IEEE*,
Domaniç Lavery, *Member, IEEE*, Polina Bayvel, *Fellow, IEEE* and Georgios Zervas, *Member, IEEE*

***Abstract*—We introduce PULSE, a sub-μs optical circuit switched data centre network architecture controlled by distributed hardware schedulers. PULSE is a flat architecture that uses parallel passive coupler-based broadcast and select networks. We employ a novel transceiver architecture, for dynamic wavelength-timeslot selection, to achieve a reconfiguration time down to O(100ps), establishing timeslots of O(10ns). A novel scheduling algorithm that has a clock period of 2.3ns performs multiple iterations to maximize throughput, wavelength usage and reduce latency, enhancing the overall performance. In order to scale, the single-hop PULSE architecture uses sub-networks that are disjoint by using multiple transceivers for each node in 64 node racks. At the reconfiguration circuit duration (*epoch* = 120 ns), the scheduling algorithm is shown to achieve up to 93% throughput and 100% wavelength usage of 64 wavelengths, incurring an average latency that ranges from 0.7-1.2 μs with best-case 0.4 μs median and 5 μs tail latency, limited by the timeslot (20 ns) and epoch size (120 ns). We show how the 4096-node PULSE architecture allows up to 260k optical channels to be re-used across sub-networks achieving a capacity of 25.6 Pbps with an energy consumption of 82 pJ/bit when using coherent receiver.**

***Index Terms*—Optical interconnections, Circuit switching (communication systems), Scheduling, Star coupler network, WDM, TDM, SDM, fast tunable transceivers, fast network reconfiguration**

I. Introduction

THE rapid increase in the rate of intra-data center traffic, due to the growth of data services, requires the supportive growth of resources within data center networks (DCNs). Cisco's Visual Networking Index (VNI) suggests that 73% of DCN traffic is internal within the data center network [1]. According to Google, the demand for bandwidth in data centers doubles every 12-15 months [2]. Intel anticipates that 70-80% of their computing and storage systems will be deployed into data centers by 2025 [3]. While data center operators are being forced to scale their computing resources they are also required to maintain low costs and power consumption. In 2015, total power consumption of DCNs worldwide was 416.2 terawatt hours, while the total power consumption in the UK was approximately 300 terawatt hours [4]. In some cases, energy use is a substantial cost relative to the IT hardware itself [5]. Moreover, by 2021, about 95% of all data center traffic will originate from the cloud [6]. In bursty, cloud based applications, 90% of packets have a size of less than 576 bytes [7]; smaller packets require faster switching. However, current electronic packet switched networks have long tail latencies of several hundred milliseconds; orders of magnitude higher than median latency, degrading application performance [8]. Hence, there is a need for networks, which reconfigure at nanosecond timescales and ensure low and deterministic tail latency and high network throughput. Hence, we propose the use of optically switched networks which can be tailored to showcase the aforementioned features.

Optical switch technologies, including Arrayed Waveguide Grating Routers (AWGRs) and star-coupler based switches, have been proven to scale to port counts as high as 512 or 1024 ports [9], [10]. With WDM, TDM and advanced optical modulation techniques, optical transceivers are also able to unlock and support higher data transmission rate (or bit rate) per port [11]. However, a major challenge faced when scaling an optical switch is the scalability of the central scheduler [12]. Hence, many optical switch technologies resolve to software-based scheduling to simplify switch configuration [9], [10], [13], [14]. Nevertheless, software based control systems take milliseconds to compute switch configurations. To tackle this, researchers have proposed optical switch solutions with distributed scheduling for simplified control. However, these proposals tend to elevate data plane complexity [15]–[17]. Recent optical packet switching technologies with fast switching control have also been shown to have scaling restrictions as they require complex optical data plane architectures, multiple optical components [18] and packet management techniques to realize electronic packet switch functionalities [19].

Previously, we proposed a transceiver-based optical circuit switched (OCS) network with a passive star-coupler core [20], scalable to 1000 ports [21] that used a centralized scheduler to reconfigure the network by defining wavelength and timeslots at the transceivers [22]. However, the broadcast and select network suffered from resource wastage as only W(=80) wavelengths were used for N(=1000) servers (network efficiency = 8%) and had a long circuit duration (*epoch*) of 2 μs, well above the aforementioned target of nanosecond timescales. In this paper, we propose PULSE, a novel scalable OCS architecture that supports nanosecond speed reconfiguration time, while enabling network and capacity scalability with the help of independent distributed hardware schedulers with high network efficiency ($W = N$). We introduce novel transceiver architectures that enable faster circuit reconfiguration time and evaluate the network energy consumption of the proposed transceiver combination. We introduce a novel scheduling algorithm that can effectively compute a new wavelength configuration (per node) for each timeslot within each *epoch* (circuit cycle duration) and limit tail latency to a few microseconds.

The PULSE architecture does not require in-network routing/switching, buffering and addressing. However, it requires ultra-fast (a) O(ns) scheduling, (b) tunable wavelength switch-

The authors are with the Department of EEE, University College London, United Kingdom (E-mail: g.zervas@ucl.ac.uk)

ing, (c) filtering, (d) distributed/scalable transport network, (e) clock and data recovery [7] and (f) O(100 ps) synchronization. Research is being carried out on several aspects of PULSE but the primary focus of this paper is to:

- propose a novel modular and distributed architecture (d) that eases control and data plane scalability (Section II).
- propose novel ultra-fast O(ps-ns) wavelength-timeslot selective transceiver architectures (b-c) (Section III).
- propose a novel hardware scheduler design (a) and evaluate its scalability (Section IV).
- review schedule performance under various epoch sizes, traffic distributions/loads to evaluate its effect on resource utilization, throughput, latency and transmitter/scheduler buffer size compared to previous work (Section V).
- study the implications of network and transceiver architecture on scalability, cost, power and latency overhead (Section V).

Section VI presents related work in the optical circuit switching field in order to identify the relevance and novelty of PULSE and we conclude in section VII.

II. OCS Network Architecture

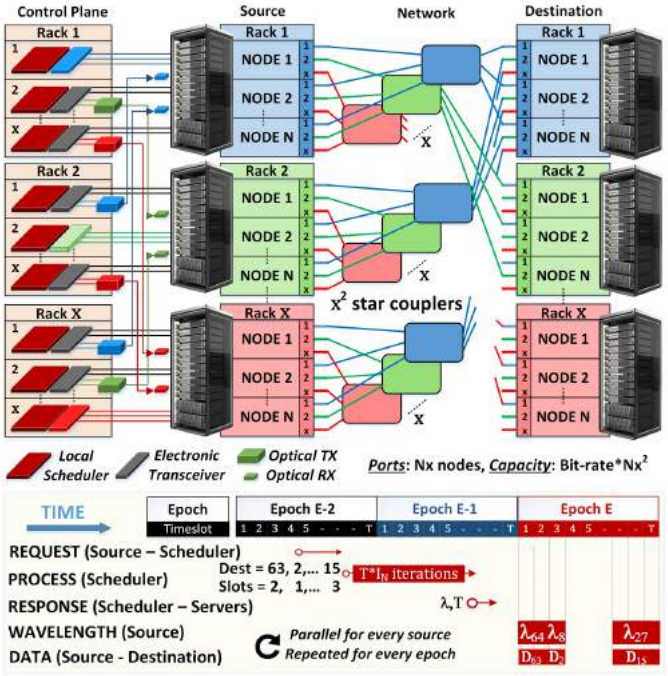

Figure 1: PULSE - Top: Parallel OCS network architecture with distributed hardware schedulers, Bottom: Control-Data handshake and dataflow.

The data plane of PULSE consists of parallel star-couplers grouped per rack, shown in the top right side of Fig. 1. Each passive star-coupler core forms a broadcast and select network [20]. There are up to x racks, where each rack contains N nodes. Each node houses up to x optical transceivers, where each transceiver connects a node to a different star coupler and thereby, a different sub-network or rack. For connection establishment and eventual data transmission across the switch, the source transmitter and the destination receiver must both tune to the same wavelength and timeslot. As shown by the top left side of Fig. 1, for every star coupler network in the data plane, the control plane has a corresponding N-node scheduler co-hosted in the same rack, which processes the requests for that particular source-destination rack pair. The parallel OCS network proposed requires x^2 N-node star couplers and schedulers. The PULSE architecture scales to support up to Nx nodes with a capacity of BNx^2, where B is the effective line-rate of each transceiver.

As shown by bottom part of Fig. 1, the communication timeline groups timeslots to form *epochs*. The scheduler also computes, for an epoch, wavelength-timeslot grants for a specific epoch. Figure 1 shows a source node sending requests to the relevant scheduler that is associated with the destinations of interest (D_{63}, D_2, D_{15}) with the number of timeslots requested (2,1,3 respectively) and timeslot number well over an epoch in advance. The scheduler performs as many iterations as it can in one epoch to compute the wavelength ($\lambda_{64}, \lambda_8, \lambda_{27}$ in Fig. 1) and timeslots for each request. Table I shows how the network, requests, racks, cables, channels and capacity scale while increasing the number of transceivers and racks ($x \in$ 4,8, 16, 32 and 64) at 64 nodes per rack ($N = 64$).

Table I: PULSE: Scalability, Capacity, Complexity at $N = 64$

Network Parameters	**Transceivers per node (x)**				
	4	**8**	**16**	**32**	**64**
Total nodes	256	512	1024	2048	4096
Req/node/epoch (R)	24	48	96	192	384
Racks (x)	4	8	16	32	64
Sub-stars (x^2)	16	64	256	1024	4096
Cables ($N \times x^2 \times 4$)	4096	16384	65536	0.26M	1.04M
Channels (Wx^2)	1024	4096	16384	65536	0.26M
Max capacity (Tbps)	100	400	1598	6394	25575

The spatial division multiplexing (SDM) created by the parallel and distributed star-couplers enables the re-use of wavelengths. In other words, the same wavelength can be re-used in another star-coupler. The complete independent nature of individual sub-networks means that local schedulers have no dependency on the traffic or resource usage faced by other stars. The synchronization problem is also reduced to a local sub-network and not required at a global level. Each 64-port sub-star uses 64 wavelengths and the SDM enables 4096 parallel uses of wavelengths allowing a total of 0.26M channels and a network with an enhanced capacity of $\sim$ 25.6 Pbps, accounting for tuning overhead. Each sub-network creates shareable bandwidth resources between two racks. The overall network bandwidth (between all racks) is indicated by 'Max capacity' in Table II. However, at any instant, the maximum node-to-node capacity is 100 Gbps.

As shown in table II, each node uses up to 6.4 Tbps. PULSE is a network solution, where each node can be a high-performance computational resource in a heterogeneous cloud DCNs - CPU, GPU, TPU, HBM (>1 Tbps) or ToRs. Resources like HBMs and GPUs [23] require more than 1 Tbps bandwidth. Large multi-core chips are also being explored to support fast AI calculations [24]. There has been considerable growth in the performance of GPUs, nearly 1.5 times a year [25] and reaching 1.5 Tbps by 2020 [23]. Although network

and computational processing are capable of handling high throughput (100 Tbps), NIC or data ingest capacity constraints (100 Gbps) create bottlenecks, leading to inefficient systems that constraint applications to operate locally and degrade the overall application performance. Hence, the FastNICs program initiated by DARPA [26] aims to boost network NIC and stack capacity by 100 times to accelerate distributed application performance and close the gap between processing and network capability. Hence, it is expected that in the future end-nodes will support high bandwidths (6.4 Tbps). Today's expensive network switches are over-designed and under-utilized [27]. PULSE intends to maximize bandwidth utilization even if high capacities are generated at the nodes.

III. Optical Circuit Switch Elements

A. Transceiver technology

In this work, we propose the use of fast wavelength and timeslot selectable transceivers, as shown in Fig. 3. For the purpose of evaluating latency, cost, power and throughput, we assume a line-rate of 100 Gbps per wavelength. The line-rate could be achieved by adopting PAM-4 (56 Gbaud) for direct-detect and DP-QPSK (4×28Gbaud) for coherent systems; though the wavelength-timeslot switching is format/rate flexible. At 100 Gbps, this system targets the transfer of 250 bytes per 20 ns time-slot, which corresponds to the overall median packet size across various data center workflows [28]. As the globally synchronized optical switch only needs to carry the bare payload of the packet, the effective packet length is slightly more than an average Ethernet packet. In addition, prior work has experimentally demonstrated fast time-slot switching with a 1-bit guard band, minimizing any TDM overhead [29]. At each transmitter, smaller packets ($\ll$250 bytes) to the same destination are aggregated for effective slot usage.

1) Prior work: Dynamic wavelength assignment (using software algorithms) with all processing and buffering at the edge was proposed in [11], [30]. The work in [21], [29] proposed each node to be equipped with a tunable DS-DBR laser and a coherent receiver with an independently tunable DS-DBR local oscillator laser. This enables fast wavelength selectivity, O(10ns), at both the transmitter and receiver, and the high sensitivity of the coherent receiver also enables scalability in the data plane since it allows for a larger system loss budget and thus a higher port count star-coupler [21], [29]. Although a 1000-port star coupler network can be created, experiments have shown that only up to 89 wavelengths can be supported by the laser hardware positioned on a 50 GHz ITU grid within the optical C-band to ensure minimal crosstalk between channels ($W \ll N$). Extending the same grid to the L-band could allow the use of 160 wavelengths. However, Fig. 2 shows the growth of this tuning time when using an extrapolated regression model with greater than 99.9% confidence interval; this predicts that a 160-wavelength system takes more than 1μs to tune all transceiver pairs. The large tuning time has a direct impact on switch latency and tail latencies, which degrades network performance, setting a limit on the wavelength channels allowed, assuming the tunable DS-DBR lasers are used at the transceivers. This performance degradation has an impact on the scalability of the OCS network in terms of capacity, limiting $W \ll N$. Regardless of the number of wavelength channels used, the large tuning time would require a dedicated synchronized tuning time prior to every epoch, which lasted in O(1μs) to minimize overhead.

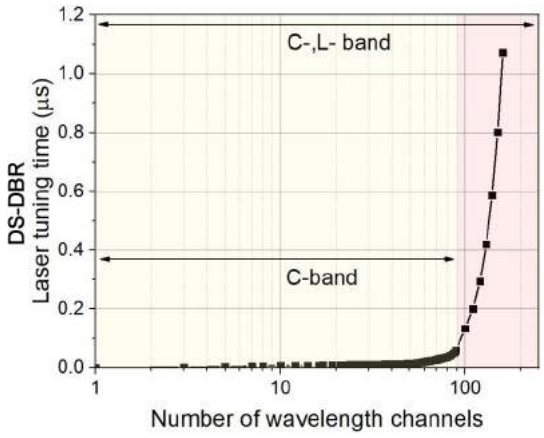

Figure 2: DS-DBR laser tuning time prediction versus the number of available wavelength channels with 99.9% regression on [21].

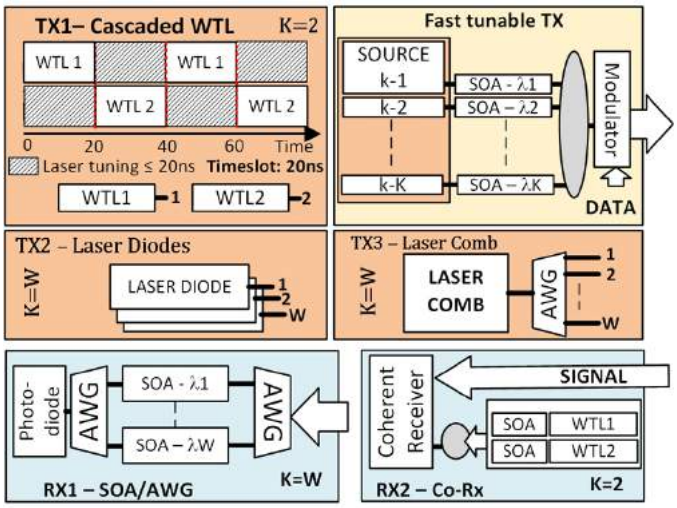

Figure 3: Transceiver options for PULSE: TX1-Cascaded WTL, TX2-Laser Diodes, TX3-Laser Comb, RX1-SOA/AWG, RX2-Coherent Rx.

2) Transmitter options: Following the previous argument, we explore methods of approaching a larger number of wavelength channels (W), while making sure that optical transmitters do not impede switching times. With that aim, we consider 3 WDM transmitter architectures that have the potential to achieve sub-nanosecond switching. Electro-optical amplifier switch based on chip-on-carrier SOA with tunable lasers were experimentally demonstrated to achieve switching times of up to 115 ps by [31]. Here, we use k such SOAs at the transceiver, in order to enable reconfiguration at a faster rate of the O(10ns), connected to one of three laser source options that we propose as shown in Fig. 3. In our transceiver

proposal, the first transmitter laser source option (TX1) is to employ k widely tunable lasers (WTLs). The WTL could be the DS-DBR lasers described in the previous sub-section or SG-DBR lasers in a cascaded fashion, connected to k SOA gates, as shown in Fig. 3. However, the WTL is required to tune to a new wavelength within 20 ns. Modulated grating Y-branch (MG-Y) lasers [32] can be used as WTLs to achieve fast wavelength selection within 20 ns. The work in [33] has shown SG-DBRs to achieve wavelength tuning in 5 ns. Prior experiments have also shown that 67% of transitions between any DS-DBR equipped transceiver pair of the 89 available wavelength channels (59 channels) complete tuning within 20 ns [21]; work is underway to improve this to support 64 channels. As shown in Fig. 3 (TX1), $k = 2$ cascaded WTLs (and SOAs) are required at the transmitter and receiver to create 20 ns timeslots. This will further reduce cost and power consumption. The star-coupler core has a loss of $-3log_2N$ dB, which corresponds to -18 dB for a 64-port network. The SOAs used at the transmitter (and receiver option 1) compensate for this loss by providing a +10 to +15 dB amplification [31] (per SOA per path). Coherent receiver (receiver option 2) are highly sensitive (-21 dB with BER of 10^{12}) [21] and can accommodate a larger split. In this paper, we aim to increase the network transceiver efficiency to 100% by increasing the number of channels to the number of nodes in the sub-network ($W = N$). Hence, the paper will investigate the network performance for (N=) 64-node racks and (x=)16 racks, scaling to 1024 nodes. The second transmitter laser source option (TX2) is the use of k(=W) VCSEL laser diodes [34], one for each wavelength, at the source connected to k SOA gates as shown in Fig 3. The third transmitter laser source option (TX3) is to use a laser comb that generates k(=W) wavelength sources connected k SOA gates [35].

3) Receiver options: We propose two options for the receivers, as shown at the bottom of Fig. 3. The first receiver option (RX1) contains an array of W SOAs surrounded by AWGRs, followed by a direct detection photodiode. The selection of the SOA allows the fast wavelength selection (O(100 ps)) at the receiver. The disadvantage of such a receiver is the requirement of many SOA gates, which has an impact on overall power consumption. The second receiver option (RX2) contains k(=3) DS-DBR tunable transmitters with SOA gates, as at the transmitter, which serve as local oscillators (LOs) for a coherent receiver.

B. Control Plane

Each node sends requests to the scheduler a *few* epochs in advance (depending on the dominant propagation delay) and awaits response before reconfiguring transceiver wavelengths for the subsequent epoch. A control sub-network requires all N nodes to be connected to the local scheduler. Requests sent from each of the nodes have the following structure: requested destination (6 bits), slot size (up to 5 bits for 600 ns epoch) and epoch stamps (8 bits) to identify the requests. Once received, the central scheduler stores requests from all nodes in a 1.2 kB buffer, which can store up to R(=6 requests) per node per epoch. The control request communication needs to communicate 19 bits within a timeslot (20 ns), and hence, requiring a 1 Gbps link. While running up to I iterations to process the requests, the scheduler stores the wavelength-timeslot pair grant information in a second buffer of 1.2 kB. Once the schedule is computed for an entire epoch, the wavelength-timeslot pair grant information is communicated. Each grant has the following structure: TX/RX wavelength (6 bits each), destination node (6 bits), valid bit (1 bit) 19 bits for 64-port network. Hence, the grant communication also requires a 1 Gbps transceiver link to send 19 bits in every timeslot (20 ns). As shown in Fig. 1, the control plane of each sub-network is co-hosted within every source rack to keep the request-response handshake propagation delay to a known minimal constant ($\approx$ 30 ns). Regardless of where the destination rack is hosted, the control plane propagation delay is a constant, as will be discussed further in section V.

IV. Hardware Scheduler Algorithms

In this section, we describe two hardware implementable scheduling algorithms, which can use any of the fast tunable transceiver options shown in Fig. 3. To meet the strict timing requirements, we adopt a parallel design that considers at least N requests, one from each of the N nodes. Although parallelism speeds up the scheduling process, it also creates contention for wavelength and timeslot resources. We use round-robin arbiters in our hardware design to ensure the fair selection of up to N contention free source-destination requests, with unique source and destination ports, per clock cycle. The selected contention-free node-pairs are assigned wavelengths (WDM) and timeslots (TDM) in parallel. The scheduler aims to perform I iterations within one epoch to maximize throughput. The number of iterations that the scheduling algorithm can perform is limited by the epoch size ($E \in 120, 360, 600$ns) and the clock period of the hardware (clk=2.3 ns for 64-port scheduler). Requiring b cycles to boot up ($b = 3$ for epoch-level scheduling, $b = 4$ for slot-level scheduling), the maximum number of scheduler iterations is $I = \lfloor E/\text{clk} \rfloor - b$. After I iterations, failed requests are buffered and are given a chance to retry in subsequent epochs.

We propose two distinct algorithms for resource (wavelength and timeslot) allocation: epoch-level and slot-level scheduling algorithms. As the name suggests, the epoch-level scheduling algorithm aims to tune the transceiver wavelengths *once* at the beginning of every epoch. However, in slot-level scheduling algorithm, wavelength is tuned *once* between each timeslot. The tuning overhead is higher for slot-level scheduling and it is taken into account while discussing throughput results in section V. From a scheduling perspective, the slot-level scheduling algorithm unlocks a new level of flexibility by reducing contradictions created by parallel wavelength assignments. Although both the epoch-level and slot-level scheduling algorithms use similar hardware elements, the major difference between them lies in the strategy used to allocate resources, which will be discussed in this section.

A. Hardware Scheduler Design

The hardware logical elements used to synthesise and implement the scheduling algorithms are discussed in sub-

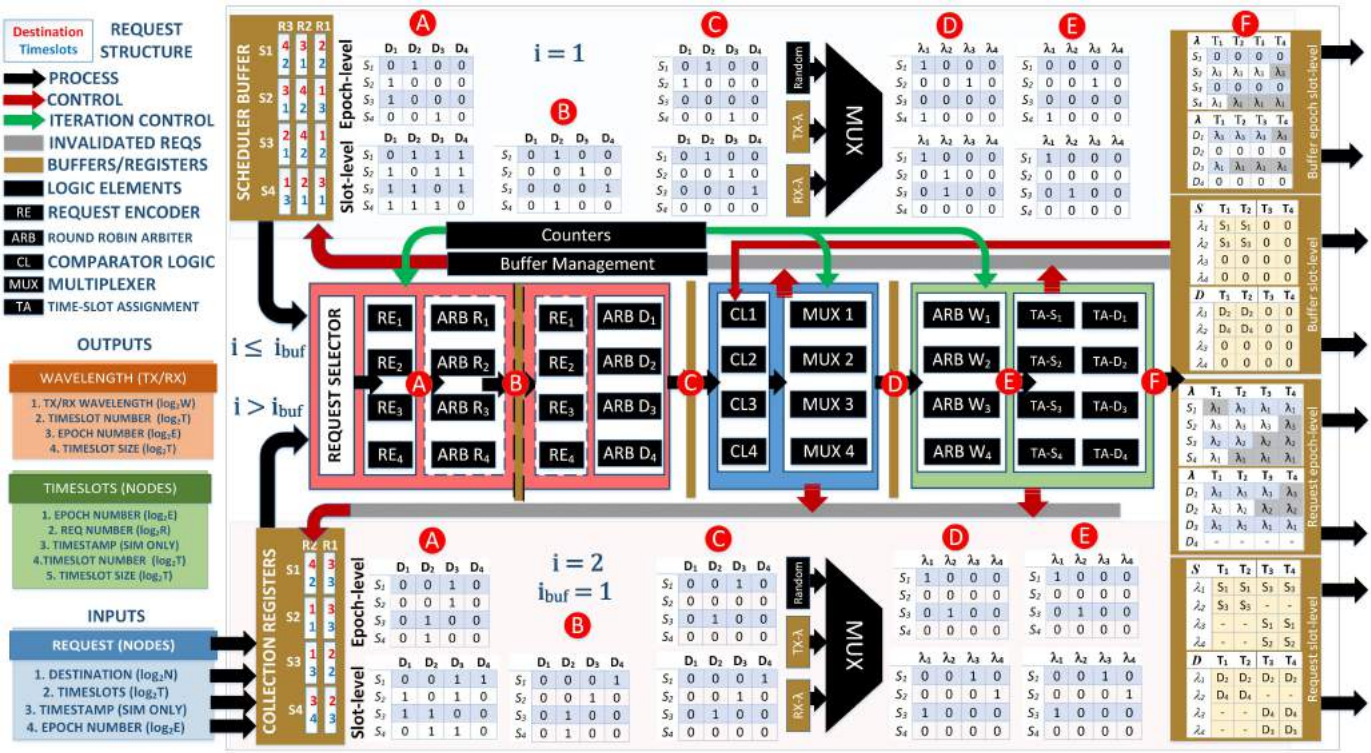

Figure 4: Epoch/Slot-level algorithm: 4-port central scheduler showcasing three hardware stages with a 4-port scheduling example showing two iterations, first handling requests from buffer (top), then from the node (bottom) (when $i_{buf} = 1$).

section. The dominant element used in the realization of the scheduling algorithm is the round-robin arbiter shown in Fig. 4, as they are scalable [22]. The critical path lies in the carry chain of the arbiters, like in digital adders. Using optimal length carry look-ahead generators can provide high-speed arbitration by shortening the critical path. We also show how the logical hardware components work together to efficiently allocate resources by using an example showcased in Fig. 4. Figure 4 shows the epoch-level and slot-level scheduling for a 4-port scheduler and the iterations dealing with buffer (i=1 shown by the top half) and dealing with requests from node (i=2 shown by the bottom half).

1) Node Contention Resolution (NCR): This module is shown by the red block in Fig. 4. In epoch-level scheduling, the NCR module has only one pipeline stage containing N parallel N-bit request encoders (RE) and N parallel round-robin arbiters (ARB D). The hardware elements in Fig. 4 that have dotted outline do not exist in the epoch-level scheduling, but only exist in the slot-level scheduling. In epoch level scheduling, every clock cycle deals with one request per source. Each RE converts the destination requested into a one-hot binary vector to create a compatible input for the round-robin arbiter that follows. The matrices labelled as epoch-level in Fig. 4 and denoted by 'A' show the encoding of the *first* requests from each source (R_1), both in the buffer iteration (epoch-level matrix at the top of figure, $i = 1$) and the node requests (epoch-level matrix at the bottom of figure, $i = 2$), to form $N \times N$-bit matrices. Each of the N round-robin arbiters deals with resolving contention for one destination (ARB D). All N sources for a particular destination are given as an input to the destination round-robin arbiters and only one request is successful per destination per iteration (labelled as epoch-level in Fig. 4, denoted by 'C').

In contrast, the NCR module in the slot-level scheduling algorithm has two pipeline stages. The first stage has N parallel N-bit RE and N *source* round-robin arbiters (ARB R). The second stage, containing the same hardware components as the first stage, has N parallel N-bit RE and N *destination* round-robin arbiters (ARB D). Notice that the matrices formed after encoding in Fig. 4, denoted by A, contains all destination requests ($R1, R2$ and $R3$). This is the case in both the requests from the buffer (slot-level matrix at the top of figure, i=1) and the new requests from the nodes (slot-level matrix at the bottom of figure, i=2). Considering multiple requests from the same source, the slot-level scheduler requires both source arbiters, ARB R (source contentions are resolved in slot-level matrix 'B'), and destination arbiters, ARB D (destination contentions are resolved in slot-level matrix 'C'). The pipelining of arbiters in slot-level scheduling, however, creates repeated grants. This is dealt with using a feed-forward control; if requests are repeated or have already been successful in previous iteration they are cancelled in the second stage.

Although Fig. 4 does not show a decoder, each pipeline stage in the NCR module contains a decoder in both scheduling algorithms. The decoders help to minimize the register size after each stage to $log_2 N$ bits per source to store only the winning destination, instead of N bits per source.

2) Wavelength Decision (WD): The wavelength decision module (WD), shown by the blue block in Fig. 4, is a pipeline stage that has the same hardware and functionality

in both epoch-level and slot-level scheduling algorithms. This module/stage is composed of a parallel comparator logic (CL) blocks and parallel multiplexers (MUX), one per source, and the select signal of the MUX is tied to the resource registers at the last stage. This stage works on contention-free node-pairs and uses information from registers to perform a parallel check on node-pairs, by reading previous resource allocations of both the source and the destination of interest in a particular iteration. This is shown by the red line feeding back from registers to CL in Fig. 4. If contending wavelengths have already been assigned to the node pair of interest, the request is invalidated for the current epoch (shown by the red arrow going to the shaded gray lines going to request collect registers to buffer registers). If either only the transmitter or only the receiver has been assigned a wavelength, then that particular wavelength is selected (shown by the MUX in the diagram), subject to time-slot availability. If there is no wavelength assignment history for the transmitter and receiver and resources are available, a random wavelength is assigned, subject to time-slot availability. In the hardware, a ROM is implemented to hold N arrays of $T \times log_2 W$ pseudo-random wavelengths (5-25 bytes per source). The selection of wavelengths in this stage does not guarantee a successful grant. The selection of same wavelengths in parallel can create contention in the parallel resource allocation process. Hence, the selected wavelength assignments and the node-pairs are stored in registers and forwarded to the next stage. Again, a decoder is used in both epoch-level and slot-level scheduling algorithm to store the wavelength decision in $log_2 W$ bits per source-destination pair (in addition to the $log_2 N$ bits for destination).

3) Resource allocation or Wavelength Contention Resolution (WCR): The third module, wavelength contention resolution (WCR), is shown by the green block shown in Fig. 4. Both epoch-level and slot-level scheduling algorithms contain this pipeline stage. Firstly, this stage contains request encoders (not shown in figure), which make N parallel W-bit one-hot binary vector that translate the wavelength decisions into compatible inputs for N parallel round-robin arbiters. The formation of this source(-destination pair) and wavelength matrix is denoted as 'D' in Fig. 4 for both epoch-level and slot-level scheduling algorithms. Secondly, this module uses W parallel N-bit round robin arbiters (ARB W) to resolve contention between wavelength assignments ('E' in Fig. 4). Up to W winning grants are generated in parallel by the arbiters. Thirdly, this module contains timeslot allocators for both the source (TA-S) and destination (TA-D).

The encoders and arbiters have the same functionality in both algorithms. However, the epoch-level and slot-level scheduling work differently in the timeslot allocation blocks. In epoch-level scheduling, the winning grants of the arbiters are granted as many time-slots as they have requested, subject to availability. If only fewer slots are available, then available slots are granted and the request is updated with the new slot size and buffered (shown by the red line to the buffer in Fig. 4) for processing in subsequent epochs. The final allocations in epoch-level scheduling for both for buffer (i=1-top) and node (i=2-bottom) requests and for both the source and the destination, are shown by the matrices denoted by 'F' in Fig. 4. In epoch-level scheduling a wavelength allocation means that the source must use the same wavelength for all timeslots in the epoch. Hence, in the epoch-level matrices in 'F', the timeslots that are greyed out show that the timeslot is unused, but the wavelength for that particular source/destination is locked for the entire epoch.

In slot-level scheduling, a wavelength allocation only means that the wavelength is locked for that particular timeslot. Hence, a different allocation strategy is required. The slot-level scheduling algorithm deals with this by having two iteration phases: coarse allocation and fine allocation. The first few iterations, based on T/S_{avg} (number of timeslots in epoch/average slot size requested) are handled with coarse allocation, where parallel requests are allocated as many slots as requested, provided availability (same as epoch-level but the wavelength for particular timeslots are checked in the WD stage). The matrices in 'F' in Fig. 4 show the filling of final wavelength-timeslot resource matrix by the slot-level algorithm for the source and the destination both from the buffer (i=1-top) and node (i=2-bottom) requests. After coarse allocation is done, resources are fragmented as shown by the matrices in 'F' in the bottom part of slot-level allocation. Hence, after the coarse allocation phase, later iterations of fine allocation aim to utilize these fragments. In the fine allocation phase, each timeslot is re-visited and up to one timeslot is granted per source per iteration. All requests with winning grants are marked to avoid repeating requests in future iterations (also shown by the red line to the buffer). Due to pipelining, the WD module does not have knowledge of the most recent wavelength updates and can still request contradicting wavelengths. In such cases, the requests are rejected and prevented from requesting in the current epoch (shown by red line going to the Invalid Request block). The wavelength-timeslot configuration is updated, registered and sent to the transmitters and receivers of each node and the timeslot information is sent to the sources.

4) Iteration/Buffer Management: As shown in Fig. 4, the scheduler has a buffer where failed requests are stored in order to retry in future epochs. Hence, in every epoch, a decision has to be made whether an iteration is to be used for processing requesting from the buffer or the nodes. The scheduler iterations are managed to cater for the requests from both the buffer and node every epoch. The size of request residing in the buffer (buffer size) is constantly kept in account. The iteration ratio (buffer:node) is controlled by i_{buf} as shown in Fig. 4. Since up to W grants are generated every iteration, the total buffer size is divided by W and multiplied by a buffer coefficient (R_p - usually between 1.6-2.5), which ensures minimal buffer accumulation. In epoch-level scheduling, which is a pointer based scheduling, the pointer stays fixed to a request number in the buffer until a minimum percentage of requests are granted is met. After this, the pointer shifts to the next set of requests. Once the pointer scans through all requests in the buffer or a threshold value is met, whichever happens first, the iterations are used for requests from node. In slot-level scheduling, the initial iterations deal with requests from the buffer. Once a minimum percentage of buffer requests or a threshold value is reached,

the requests from the nodes are considered.

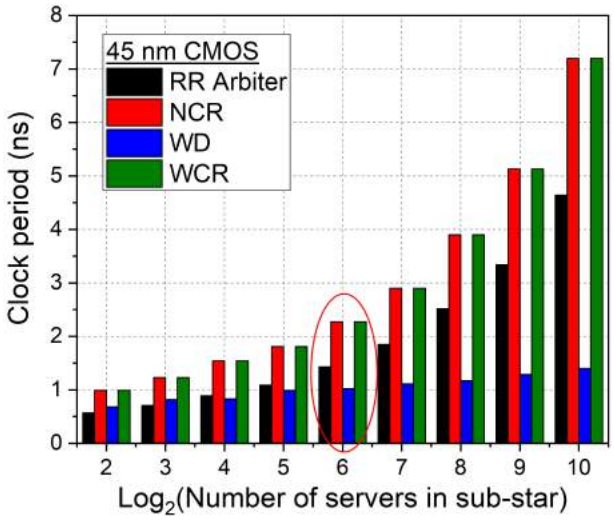

Figure 5: Critical path length of scheduler sub-modules: Node Contention Resolution (NCR), Wavelength Decision (WD) and Wavelength Contention Resolution (WCR) on 45nm CMOS OpenCell library, highlighting N=64 system for this paper.

B. Implementation

To demonstrate the functionality of our equivalent software models and hardware design, we simulated the RTL code for the hardware scheduler on Mentor Graphics Modelsim with variable scheduler request traffic inputs from MATLAB. We verified that the outputs were consistent with our software models. Once verified, the scheduler algorithm sub-modules were synthesized on ASIC, using the Synopsys tools and the 45 nm CMOS Nangate Opencell library. We used this technology as it is an open-source, standard-cell library. Figure 5 demonstrates the scalability of the sub-modules on ASIC as they are scaled to support 1000 ports on 45 nm CMOS technology. The scalability of one N:1 round robin arbiter is shown in Fig. 5. As parallelism does not significantly affect the critical path delay, $N \times N$-port and $W \times N$-port parallel round robin arbiters are used in the NCR and WCR sub-modules respectively. The two arbiter sets in the slot-level allocator are logically identical to the NCR module and are pipelined to have the same size ($N \times N$-bit arbiters). Hence, the scalability of these sub-modules are determined by the scalability of the arbiters. The wavelength decision (WD) stage, however, depends on the scalability of the checker, which uses simple scalable N:1 multiplexers to perform the checking (with a critical path of 60 ps per 2:1 multiplexer estimated by the 45nm CMOS OpenCell library).

The NCR module has the longest of the critical paths, requiring a minimum clock period of 2.3 ns for a $N = 64$ node scheduler. The area consumed by the 1000-node scheduler ASIC is 52.7 mm^2 without including the buffers or SERDES in the control plane. The scalability results show that the hardware implementation of these modules is feasible.

V. Performance Analysis

In this section, we evaluate the performance of the two hardware implementable scheduling algorithms: slot-level and epoch-level allocation. Generating demand traffic with diverse distributions, we study the effect of varying epoch sizes and request loads on throughput, latency, tail latency, transmitter/scheduler buffer size, wavelength usage and energy consumption. Firstly, software models equivalent in functionality to the hardware algorithms were modelled in MATLAB. A request traffic generator is used to feed the scheduling algorithm models with demands and the evaluation of performance is detailed in this section. The parameters and settings that were used to evaluate performance are shown in table II. As each timeslot can carry up to 250 bytes, a minimum of 120 ns is investigated, in this paper, to carry the biggest Ethernet packet.

Table II: Simulation settings

1. Algorithm		**2. Epoch size (ns)**			**3. R**			**4. TD**		
Slot	Epoch	120	360	600	2	3	6	1	2	3

All possible permutations of 1. Algorithm, 2. Epoch size, 3. Requests per node per epoch (R) and 4. Traffic Distribution (TD) were used to evaluate the scheduling performance for various input network loads, while each sub-network size supports 64 nodes/rack (with 64 port schedulers and 64 wavelength channels). However, it is important to note that at 120 ns epoch, there are only 6 timeslots/epoch and hence, some permutations are not possible at high values of TD.

A. Traffic Pattern

At every epoch, the requests generated by the source are sent to the relevant local scheduler. Each request consists of the destination node, the number of time-slots required and the origin epoch number. Firstly, to model the demand matrix generation, a uniform random distribution was used to select the destination node with a probability of $1/N$. Secondly, the average size of each request corresponds to the number of slots available in an epoch divided by the requests per node per epoch ($S_{avg} = T/R$). Up to R requests are generated by each source per epoch and a Poisson distribution is used to model the inter-arrival rate of requests. All the requests that arrive within the start of epoch are computed, else they are buffered for the next epoch. A uniform distribution of time-slot sizes of requests is used to create the slot traffic distribution (TD) with an average slot size of S_{avg}. TD1 corresponds to a single size request, where all requests are of one specific size. In TD2, a total of three size values are allowed (for example, requested timeslots per node, $S \in 1, 2, 3$); in TD3, a total of five size values are allowed (for example: $S \in 1, 2, 3, 4, 5$). To get a grasp of this double interpolation of uniform random distributions, Fig. 6 shows the number of slots requested by source-destination pair for a unique TD and R. Focusing one sub-plot, the y-axis shows the source node number of the source rack and the x-axis shows the destination node number of the destination rack. The color of the heat-map indicates how many slots are requested over a span of 2000 epochs. The generated traffic is for a 360 ns epoch at 100% input

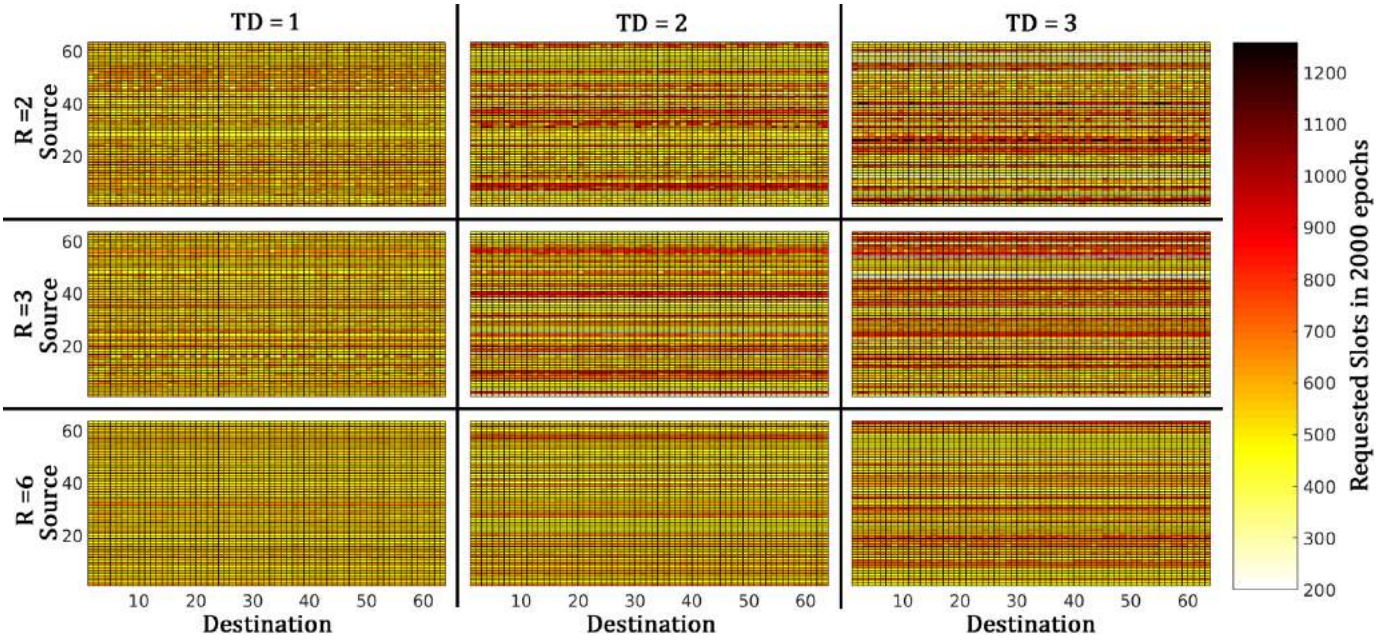

Figure 6: Active nodes: source-destination demand size for 2000 epochs for 360 ns epoch at 100% input load.

load. 100% input load corresponds to all nodes requesting up to R destinations with S_{avg} timeslots to utilize all resources (W and T) available in the sub-network.

As shown in Fig. 6, the uniform random request (on both the slot and destination) traffic generation has created a non-uniform pattern. At $R = 6$ and TD1, there is the least amount of variation in slot size between source-destination pairs. This is because as the number of request increases, the average timeslot requested is low and all requests are of the same size (TD1). As R reduces to 2, the size of each request increases creating a higher range of variance. As TD changes from 1 to 3, we can see that certain source nodes become hot in the network, which demand more resources (wavelengths-timeslots) than others. The variations are emphatic as we approach TD3 with low R. This is the type of traffic used to evaluate the performance of the scheduling algorithms.

B. Throughput Performance

The throughput of a $N = 64$ node sub-network was evaluated under varying traffic patterns and epoch sizes. The throughput of both the scheduling algorithms (slot-level and epoch-level) were evaluated for increasing input load. In Fig. 7, the effect of changing TD (column) and epoch size (row) on the throughput of both the slot-level and epoch-level scheduling algorithms is shown. Within each sub-plot, the effect of varying R is shown. The values shown take into account the relevant tuning times: 500ps for 20ns timeslot for the slot-level algorithm and 500ps for the entire epoch length for the epoch-level algorithm.

The epoch-level scheduling algorithm in all sub-plots of Fig. 7 reaches a saturation point at approximate input loads of 60, 50 and 40% for 2, 3 and 6 requests per node. In contrast, the slot-level achieves maximum throughput at 100% input load. The saturation point for the slot-level algorithm is between 85-95% while the epoch-level saturates between 35-62%. As evident, the slot-level scheduling algorithm offers an average gain in the matching performance by 32-48%, compared to epoch-level scheduling for all values of TD and epoch size.

In the epoch-level scheduling algorithm, each wavelength allocation locks the transmitter and receiver pair to that wavelength for the entire epoch. This increases blocking probability, as future iterations have to work around this *locking*. In contrast, the slot-level allocation has the flexibility of changing the wavelength every timeslot. This flexibility is permitted by the architecture of PULSE's transceiver and network. The variance in throughput in the epoch-level scheduling between $R = 2$ and $R = 6$ also ranges from 18-30%. However, this variation is contained in the slot-level scheduling to less than 5%, showing the algorithm to be more tolerant to variation in request volume. As the number of requests per node increases, the throughput in both the slot-level and epoch-level algorithm decreases.

C. Wavelength usage

Fig. 8 shows the average percentage of wavelength channels ($W = N = 64$ channels) used to achieve the matching for epoch-level and slot-level scheduling algorithms for increasing input loads in a 360 ns epoch. For all values of TD, the maximum wavelength usage in the epoch-level scheduling algorithm is 49-62%. In contrast, the slot-level scheduling algorithm achieves a maximum wavelength usage of 97-100%. In both scheduling algorithms, the input load at which the saturation point is reached in the epoch-level algorithm matches the input load at which usage of resources also saturates. At 100% input load and low requests/node ($R = 2$), that is low congestion, the saturation point is at 50-60% load. At the best performance, the epoch-level algorithm is able to use 32-38 wavelength channels every epoch out of 64. However, at 100%

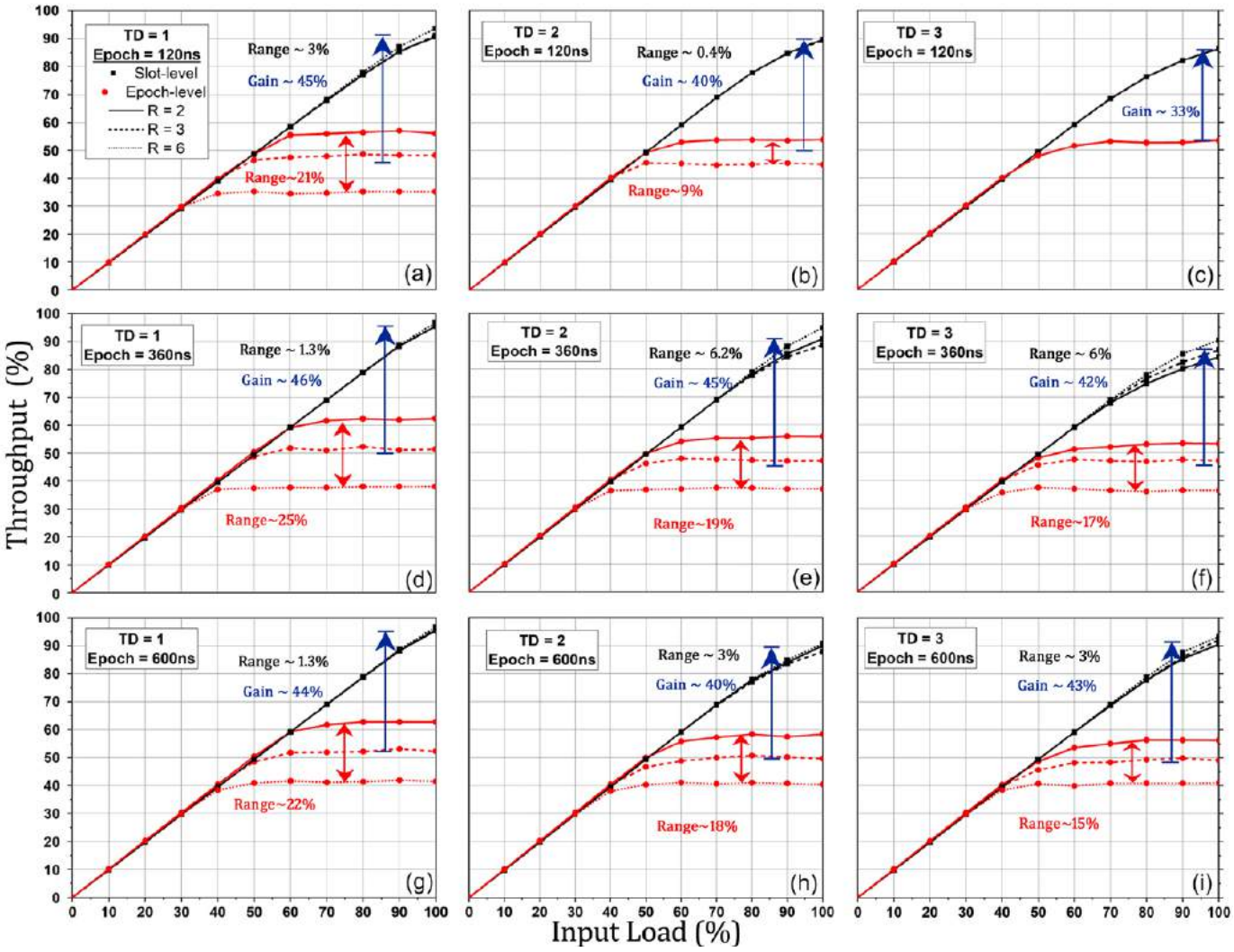

Figure 7: Scheduler throughput vs input load for varying values of R, epoch sizes, TD.

input load, the slot-level scheduling algorithm saturates very close to 100%, showcasing a network that utilizes up to 64 wavelength channels.

D. Average Latency and Transmit Buffer

To measure the latency, each request was marked with a time-stamp when generated. The scheduler has a buffer that stores failed requests in the current epoch to retry the same requests in future epochs. Once successfully and completely

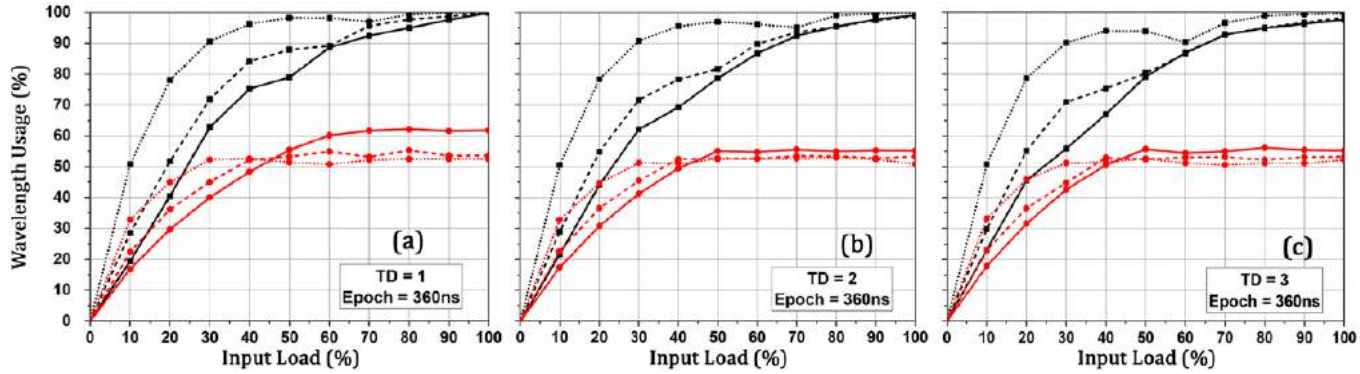

Figure 8: Wavelength usage (out of 64 wavelengths per star coupler) vs Input load for varying R, traffic distributions: (a) TD1, (b) TD2, (c) TD3 in a 360 ns epoch.

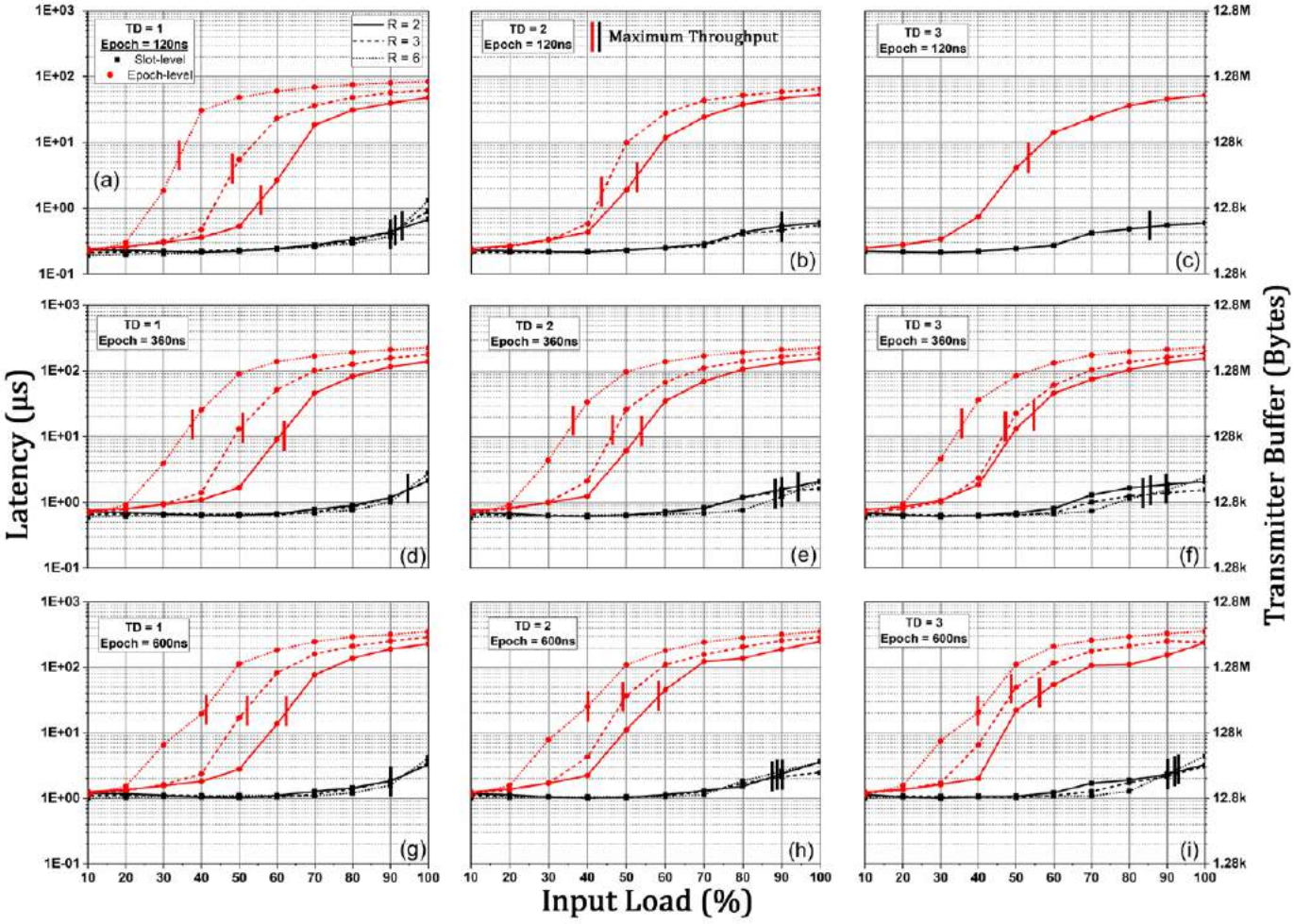

Figure 9: Average end-to-end latency and TX buffer size vs Input load for varying values of R, epoch sizes, TD.

(all requested slots) granted, another time-stamp is created to mark when the request will translate into communication in the data plane and reach the destination. The difference in the time-stamps is taken for each request to get the scheduling latency distribution over multiple (2000) epochs.

Figure 9 shows the end-to-end overall average latency taken for packets over 2000 epochs when the network is scheduled using epoch-level and slot-level algorithms for increasing input load, while varying values of requests per node (R within plot), epoch sizes (row) and traffic distributions TD (column). Each sub-plot also shows the load where maximum throughput is achieved (saturation point).

The propagation delay and tuning overhead are not considered in these measurements; they are discussed at the end of this section. The high throughput achieved by the slot-level algorithm means that many requests are granted without buffering. Hence, the slot-level scheduling algorithm is expected to have a significantly lower latency compared to the epoch-level scheduling, with the minimum latency corresponding to the epoch size.

The epoch size and R have a greater impact on the latency compared to the TD value. As expected, Fig. 9(a-c) shows that, at 120ns epochs, slot-level scheduling has an average latency of 500-600ns at around 90-94% saturation load, dependent on R, while the epoch-level scheduling algorithm consumes 1-7μs at 35-55% saturation load. In the epoch-level scheduling algorithm, higher requests/node (R) have a higher latency, whilst in the slot-level algorithm this variation is quite low. At 100% load for a 600 ns epoch (worst-case), the significant impact of epoch size on the latency of the scheduling algorithm is clear; the epoch-level scheduling algorithm incurs an average latency as high as 200 μs, while the slot-level scheduling algorithm incurs an average latency of 4μs. Hence, the smaller the epoch size, the lower the latency. At 100% input load, the latency reduction offered by the slot-level compared to epoch-level scheduling is 40-80 times, when using a 120ns epoch.

Latency has a direct impact on the average transmitter buffer size that is required at the transmitter, as shown by the second y-axis on Fig. 9. While requests are buffered in the scheduler, the data packets are buffered at the source where they await the grant from the scheduler. An average packet size of 250 bytes is assumed for every time-slot (20ns) delay. Just as latency is reduced by an order of magnitude by the slot-level algorithm relative to the epoch-level scheduling algorithm, the buffer size is also reduced by an order of magnitude. Assuming a 100% input load, the average buffer size required for a 120ns slot-level scheduling algorithm is 15.36kB, relative to the 1.09MB buffer size demanded by the epoch-level scheduling algorithm.

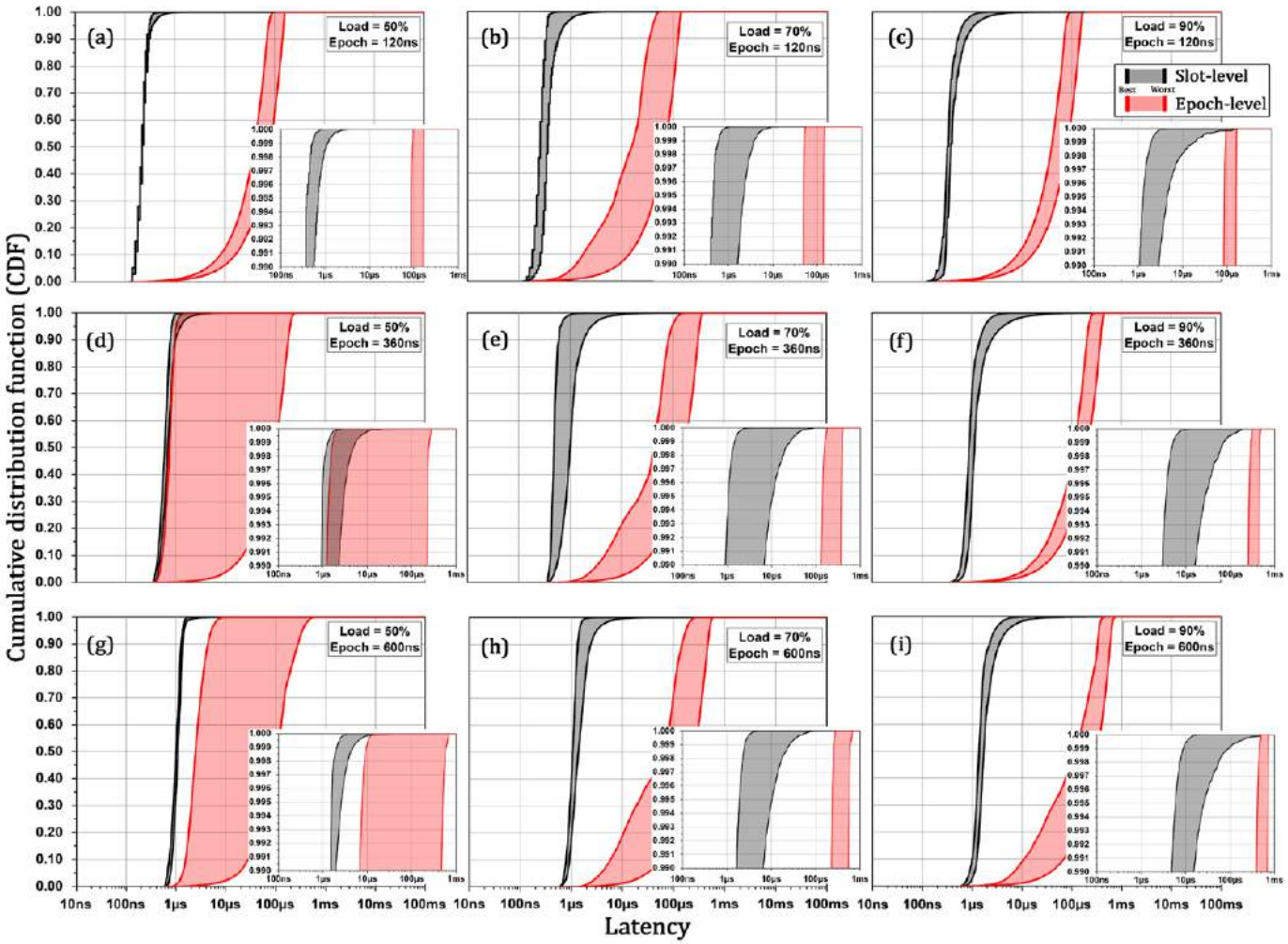

Figure 10: Scheduler latency CDF distribution (inset: tail) for varying values of R, epoch sizes, TD.

E. Latency Distribution

All requests with a successful grant (contain both timestamps from the time it was generated to the time it was granted) are considered for these latency measurements. For 2000 epochs, considering all N-ports and R requests/node/epoch, a range of successful requests came out with grants. In Fig. 10(a-i), we show the distribution of latency using a cumulative distributed function (CDF) for different epoch lengths ($\in 120, 360, 600$ns) and input loads ($\in 50, 70, 90\%$). Each row in Fig. 10 shows the scheduling behaviour under different epoch sizes. The curves clearly show a shift to the right and hence, an increase in both the median latency and tail latency. The proportion of this shift in average, median and tail latency is directly proportional to that of the epoch size. When epoch sizes are increased 3 times (360 ns) or 5 times (600 ns), the latency also increases by the same factor. Within each figure, we show the best and worst case for varying values of R and TD to showcase how it affects the latency for both the slot-level and epoch-level algorithm. The inset in each Fig. 10(a-i) shows the behaviour of the tail end of the CDF showing the packets that are buffered for several epochs. This is important to consider because the performance of applications in current data center networks are limited by the tail latency; applications have to wait for hundreds of milliseconds and hang at very high workloads [8].

The slot-level algorithm can achieve unsaturated throughput above 90%, while epoch-level algorithms can have their saturation point between 35-62%. This means that the red curves represented in Fig. 10 can only show the best case with many requests still remaining in the buffer beyond the saturation point. The slot-level algorithm shows a two orders of magnitude lower worst-case median and tail latency compared to epoch-level algorithm for all values of input load. In a 120 ns epoch in Fig. 10(a-c), the slot-level algorithm achieves a best-case median and tail latency of 0.4 μs and a tail latency of 5 μs respectively, compared to 35 μs median and 99 μs achieved by epoch-level algorithm. The worst-case tail in the slot-level algorithm at 90% input load is at around 120 μs. This corresponds to latency of almost 1000 epochs and it is mainly caused by few small packets (<0.05%) at high load.

F. Summary

In this sub-section, we summarize the results showcased in the previous sections at the saturation input loads; the load at which maximum throughput is reached. In Fig. 11(a-c),

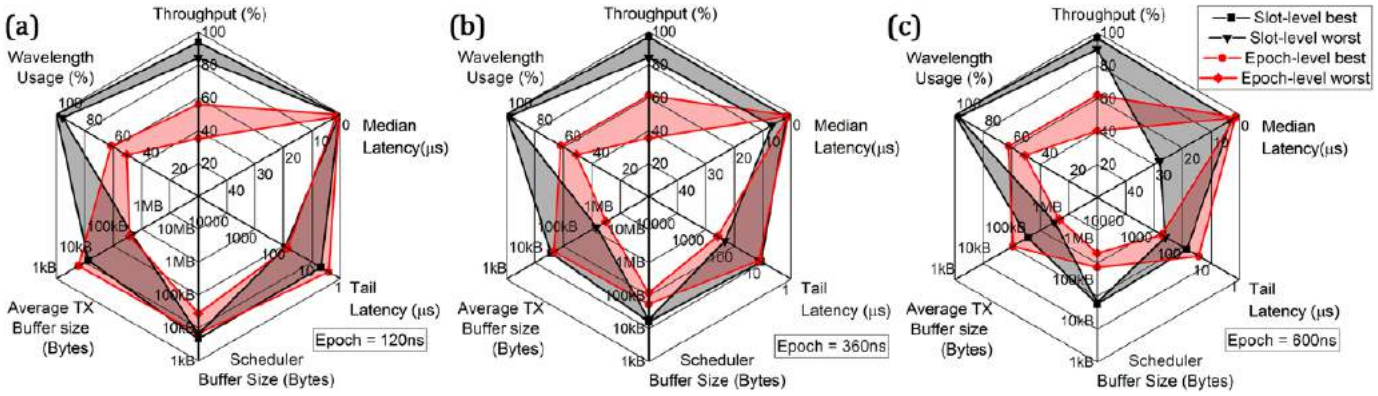

Figure 11: Radar plot - overall performance of epoch-level and slot-level scheduling algorithms at maximum operable load.

we show a radar plot showing six different axes: throughput, median latency, tail latency, average scheduler buffer size, average transmitter buffer size and wavelength usage, for 120 ns, 360 ns and 600 ns epoch sizes. For all values of TD and R, the best and worst values for identified to make this plot. A curve with a large radius or opening shows that the algorithm has increased efficiency with high throughput and maximal wavelength usage, low latency and buffer size. The best and the worst values are shown in all axes to show the variance or the range of change. Fig. 11 clearly shows how the slot-level algorithm performs better than the epoch-level scheduling algorithm. The throughput and wavelength usage of slot-level algorithm is above 90% and tolerant to changes in epoch size compared to epoch-level algorithm 35-62%. The variance in median latency of slot-level scheduling is lowest for a 120ns epoch, compared to epoch-level scheduling. Maintaining a tolerant buffer size for the scheduler and transmitter, the slot-level scheduling has a better buffer management system.

G. Scalability, Power and Latency overhead

The previous sub-sections focused purely on scheduling latency. The modulation, serialization, transceiver latency and the propagation latency were not accounted for. Hence, in this sub-section, we highlight the latency overheads that exist within the network. Fig. 12(a) shows the latency overhead for intra-rack, inter-rack or end-of-rack communication in PULSE, assuming lengths (L) of 3m, 20m and 100m respectively. Each of these links in the data plane corresponds to fiber runs of

Table III: Component count, power assumptions per TX/RX

Device	TX1	TX2	TX3	RX1	RX2	Power per Unit (mW)
SOA(on) [31]	1	1	1	1	1	405
Comb LD [35]	0	0	1	-	-	1000
Comb-Amp [36]	0	0	1	-	-	1300
LD [34]	0	64	0	-	-	80
AWG	1	1	2	2	-	-
MOD [37]	1	1	1	-	-	1460
DS-DBR [38]	2	-	-	-	2	1000
SOA(off) [31]	1	63	63	63	1	8
CO-RX [39]	-	-	-	0	1	2000
PD [40]	-	-	-	1	0	630

L meters to the coupler and L meters to the destination. The architecture has the control plane scheduler co-located with the source node racks within a 3m reach, regardless of where the destination rack resides. The co-location of the scheduler means that the latency overhead of the control plane is a known constant and, since the data plane overhead dominates, the configuration of the network is done long before the data arrives. As shown in Fig. 12(a), a total latency overhead of 0.15μs, 0.33μs and 1.12μs are incurred when communicating intra-rack, inter-rack and end-rack. Integration of large channel bandwidth-dense transceivers as integrated SiP midboard optics (MBOs) has been shown in [41], proving the feasibility of supporting densities of 64 Gbps/mm^2 (as of 2014). In 2018, an ASIC switch with in-package optical transceiver ports was demonstrated [42]. A similar co-packaging with FPGA was reported in [43] and demonstrated at [44]. Dense

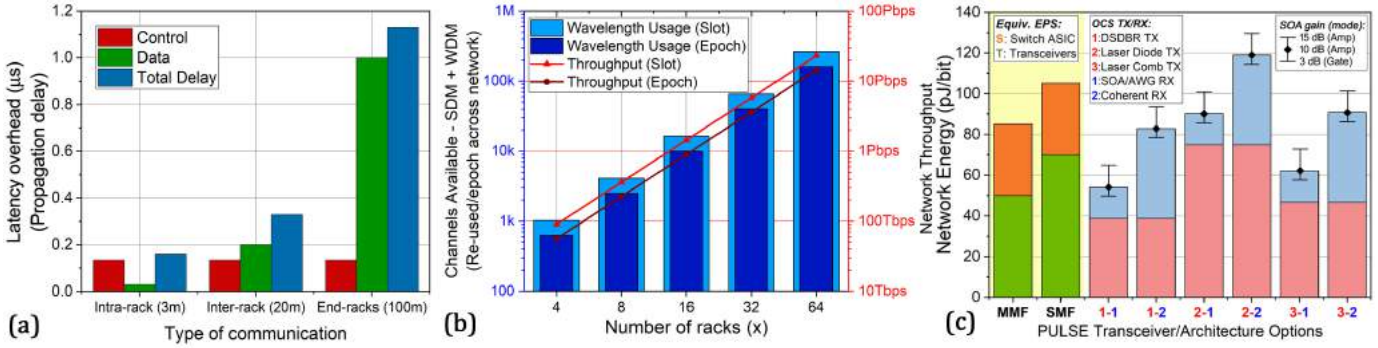

Figure 12: (a) Latency overhead: propagation delay (b) Scaling capacity with x transceivers (c) Network energy consumption.

SiP integration of transceivers can enable the accommodation of 64 transceivers on a PULSE node. Fig. 12(b) shows how PULSE can scale with (a)the number of transceivers or racks ($x \in 4, 8, 16, 32, 64$). We show that at $x = 64$, we can reuse 0.26M channels and reach a capacity of 23 Pbps. PULSE is a small-scale high bandwidth data center network that scales to support up to 4096 nodes. Novel architectures that can scale to more nodes are being explored.

With an assumption that each node is equipped with a 100 Gbps transceiver, the cost and the power are normalized against end-to-end 100 Gbps link. In Fig. 12(c), we show the network energy consumption of PULSE. The electronic architecture version of PULSE replaces the star coupler of each sub-star with an EPS switch and two transceivers. In this architecture, the power consumed when employing MMF transceivers is 85 pJ/bit; however, when using SMF transceiver systems, 105 pJ/bit is consumed per link. State-of-the-art electronic transceivers consume 3.5 W [45] per 100 GbE port while switch ASICs assume 225 W per 6.4 Tpbs [46], their overall contribution to the network is 10.5 W/path in the EPS architecture. The details of the optical components used in PULSE, the power values assumed to estimate the network energy and their references are shown in table III. As shown in table III, an SOA consumes 405 mW (90 mA) if driven to offer an amplification of 10 dB and 8 mW (12.5 mA) if driven in the absorption mode (-20 dB) [31] As shown in Fig. 12(c), the cascaded fast tunable DS-DBR lasers (TX1) requires the least number of devices and consumes lowest power (38 pJ/bit) out of all the proposed transmitter options. The power consumption of the laser diode transmitter (TX2) is as high as 75 pJ/bit because it requires W active laser diodes, one SOA in ON state and $W - 1$ in OFF state per transceiver per network. Micro-ring resonator-based laser comb generators (TX3) require laser diode (1W power consumption) [35] and low-noise, high gain amplifiers (1.3 W power consumption) [36] to compensate for the coupling, insertion and filtering losses demonstrated in [47] as well as one SOA in ON state and $W-1$ in OFF state for gating (as per TX2); a total power of 47 pJ/bit. Selecting the DS-DBR transmitter option, (1) fast wavelength tunable filters consumes a power of 54 pJ/bit (5.4 W/port) or (2) coherent receiver technology consumes a power of 82 pJ/bit (8.2 W/port). The range on the bar indicates the power consumed if both the transmitter and receiver SOAs are driven at a different gain level. Coherent receiver is the choice of receiver elected for cost analysis, as the fast wavelength tunable filters (1) require large number of components when integrating on a photonic receiver and (2) scalability increases integration complexity (more wavelengths means more SOAs to be added).

H. Cost estimation and comparison

In order to fairly compare PULSE with state-of-the-art electronic networks, the cost of deployments with equivalent bandwidth performance (6.4 Tbps), end-nodes and full bisection bandwidth is evaluated. Cost estimates are normalized to the capacity that each component supports ($/Gbps); for example, the cost of a 100 GbE transceiver cost is $3/Gbps [48] or $1/Gbps for a 100 GbE multi-mode transceiver [52]. From our analysis, the cost of the PULSE electronic network costs $8-16/Gbps. In PULSE, a flat transceiver-based architecture, the normalized cost per path is determined by the transceiver architecture while the transport layer cost is kept to a minimum of 0.04$/Gbps, assuming a 64-port coupler cost of $240 based on double the cost of a 64-port splitter in [51]. The price of the 100 G coherent receiver transceiver is dependent on the complexity of the receiver architecture and DSP required. We have assumed that the simple to complex coherent receivers are 1.5, 2 and 2.5 times the 100 GbE direct detect receiver RUC (relative unit cost) based on the report by ASTRON [53] (as coherent transceivers offer the best energy efficiency - Fig 12(c)). Using [53], we estimated the RUC for adapting the PULSE transceiver architecture has a worst case RUC is 2.01 (2x) compared to 100 GbE transceivers. This shows while employing coherent transceivers, the cost of PULSE is $4.54-7.54/Gbps, achieving 1.1-3.5 $\times$ cost efficiency compared to equivalent electronic network. PULSE only requires end-node transceiver upgrading during a network upgrade cycle whereas electronic architectures would require both $2\times$ end-node transceiver plus switch upgrades. Hence, the upgrade cost efficiency over time is low for PULSE OCS relative to the equivelent EPS architecture. The analysis in [48] shows that price of transceivers is significantly dropping every year, which would benefit a transceiver-switched architecture like PULSE. Moreover, the report in [54] predicts an annual cost reduction of coherent solutions by 15% and that beyond 16 WDM channels, they will be more cost effective. The report in [55] predicts that 800G coherent modules employed in data center networks could approach the cost of $1/Gbps by 2024.

VI. Related Work

In this section, the relevance of PULSE with respect to previously proposed optical switch, network and scheduling solutions for data centers is identified. Although the demand for data is ever growing, the pin and ASIC bandwidth of current electronic switches are approaching a limit. Hence, extensive research and development have been invested on expensive high capacity switches. As mentioned in section I, scalable optically switched networks with low deterministic median and tail latencies can be a relative game-changer for data centers in terms of power, cost and latency. However, optical packet switches (OPSs) require optical buffer/queue management, congestion control, casting and complex data exchange protocols. OPSs cannot easily replicate the range of complex methods and functionalities that current electronic

Table IV: Cost estimate of PULSE network compared with equivalent electronic DCN as of 2019

PULSE Component	$/Gbps	#/path	
		EPS	OCS
100 GE Transceiver (20m) [48]	1-3	2	-
Arista (6.4 Tbps) [49], [50]	6-10	1	-
100 GE Transceiver (Co-Rx low)	4.5	-	1
100 GE Transceiver (Co-Rx med)	6	-	
100 GE Transceiver (Co-Rx high)	7.5	-	
Star-Coupler [51]	0.04	-	1
Total ($/Gbps)		**8-16**	**4.54-7.54**

Table V: The relevance of PULSE with respect to current leading OCS Network Research

Topology	Switching Scheme	TDMA	Switch time	TDM Slot Resolution	Min. Circuit Duration	Compute time	Architecture Goal	Device(s)
Helios [56]	Hybrid EPS\OCS	No	12ms	-	O(s)	15ms	Reduce cost, power, switch ports	MEMS
OSA [57]	Circuit	No	14ms	-	O(s)	290ms	Achieve high bisection bandwidth	WSS, OSM
Rotornet [58]	Circuit, EPS ToRs	Yes	20μs	O(100μs)	O(1ms)	-	Arbiter less, throughput maximization	Rotor switch
Firefly [59]	Circuit, EPS ToRs	No	20ms	-	O(s)	60ms	Maximal matching for high throughput	LC, Galvo mirrors
REACToR [60]	Hybrid EPS/OCS	Yes	30μs	185μs	1.5ms	O(10μs)	OCS to provide EPS performance	100G OCS, 10G EPS
Mordia [61]	Circuit	Yes	15μs	95μs	O(100μs)	O(10μs)	Maximal throughput, faster reconfiguration	WSS
PULSE	Circuit	Yes	500ps	20ns	40ns	40ns	ns-speed scheduling for ultra-low latency	SOA-based transceivers

switch ASICs perform. As optical buffers do not exist, optical packet switching schemes either use power hungry optical-to-electronic/electronic-to-optical converters and use electronic buffers or fiber delay lines to support queues. In addition to this, the scalability of the control plane also has an impact on latency of the network, as identified by [12], emphasizing the need for efficient congestion management.

Nanosecond speed optical circuit switched (OCS) networks are a perfect solution to this problem as they can keep the latency low and deterministic, while keeping the complexity minimal. OCS networks eliminate queues within the switch, the associative issue of packet loss and the need for addressing. OCS offers flexible adjustment to traffic patterns as circuit establishment can last from a few nanoseconds to several hours. However, the key challenge in designing a fast OCS network is the scalability and speed of the hardware scheduler.

A. Optical Circuit Switch Solutions

Recent OCS solutions, switching schemes/techniques, their reconfiguration, computation time and circuit duration, architecture goal and the components they use are shown in table V. Micro-electro-mechanical system (MEMS) based high capacity optical switches can perform high bandwidth workload off-loading (Glimmer-glass in HELIOS architecture [56]) and reduce overall electronic switch count, cost and power consumption. However, the slow configuration time of the MEMS-based optical circuit switching, about 27ms (table V) limits their application to long-lived stable traffic; they need to work in co-ordination with electronic packet switches to cater for diverse types of bursty traffic. A faster single comb driven 2048-port MEMS switch has been proposed that can achieve a switching speed of 20μs [17]. However, a MEMS OCS still incurs substantial latency when switching small size data (e.g. 20ns data packets), making them suitable only for long-lived data flows. The OSA optical switch architecture also incurs latency in the order of milliseconds due to the high computation time it requires [57]. Although the switching time in Rotornet is reduced to tens of microseconds and computation time to zero (cyclic switching), the cycle wide reconfiguration time reduces the quality of service under realistic data center traffic patterns [58]. Firefly also suffer from high reconfiguration time requiring mirror switching or guiding [59]. REACToR [60] and Mordia [61] have a fast switching time and support TDMA; however, the long circuit duration of the order of (sub-)milliseconds would result in longer tail latencies. In PULSE, we propose a fast tunable transceiver that coordinates with SOAs to perform switching at 500ps, establishing 20ns timeslots. The circuit is computed for multiple timeslots, 120-600ns in this paper, and a hardware scheduler capable of performing $2.3 \times I_E$ iterations is shown.

B. Optical Scheduler Solutions

The total switch configuration time is defined by both the data and control plane scalability. A software-defined FPGA based approach to configure small port-count optical switches was demonstrated to take 53 ms [13]. Distributed and centralized MAC protocol based heuristics have also been proposed to control optical switches. In the 64-port POTORI (coupler based) switch, a centralized and tailored MAC protocol uses Largest First (LF) and iSLIP scheduling heuristics, which were shown to incur a latency of 10 ms above 80% workload [14]. The c-MAC control scheme in the AWG-based petabit switch architecture has an estimated latency of 5 μs (also for 64-port AWG) for offered network loads above 70% [62]. Software based scheduling solutions are orders of magnitude slower than the reconfiguration times of PULSE; hence, the need for hardware based schedulers is clear. The Data center Optical Switch (DOS) architecture uses label extractors in front of AWGRs to identify destinations, resolve contentions and configure tunable wavelength converters (TWCs) [63]. DOS assumes an unrealistic hardware clock speed of 2 GHz for its N input port, N output port arbiter elements.

In contrast to the above OCS switching technology, PULSE identifies the need for a specially designed hardware architecture that enables fast scheduling. The novel transceiver architecture in PULSE gives greater than 3 orders of magnitude faster switching time at 20ns with minimum circuit duration lasting at 120ns. PULSE scheduler algorithm exploits spatial and temporal parallelism and each iteration can grant up to 64 ($\times S_{avg}$) timeslots in fine (coarse) allocation at 435 MHz. The timeslot level allocation enables PULSE to achieve a highly tolerant throughput at 88-93%.

C. Synchronization and CDR locking

PULSE requires nanosecond resolution time-slot synchronization (each time-slot is 20 ns). Although practical realisa-

tion of this synchronisation is beyond the scope of this paper, it is still a crucial requirement for error-free communication. Prior research has shown optical fiber clock distribution to 1000-ports with jitter less than 12 ps, using mode-locked semiconductor lasers [64]. Reliant on the use of a dedicated 1 Gbps synchronization plane, the White Rabbit project can achieve a clock accuracy better than 1ns and precision better than 50 ps spanning distances over 10km [65].

Recent practical demonstrations by [7] have shown clock data recovery (CDR) locking to be achieved in < 625 ps using *phase caching*. The phase information is required to be updated only once every minute, which makes the CDR settling time a negligible overhead and hence, they are not considered in latency and throughput measurements. This removes preamble needs, although phase has to be updated once per several million epochs. This technique has been demonstrated for a 25 Gbps OOK modulation in real-time and is practical for a 100 Gbps transceiver.

VII. Conclusion

In this paper, a fast and novel transceiver-based OCS network architecture that enables switching at nanosecond timescales has been proposed. Novel fast hardware schedulers that assign resources (dynamic wavelength and timeslots) in nanoseconds for fast network reconfiguration were designed, implemented and their performance evaluated. The parallel design uses arbiters for selection of requests to grant resources. The implementation of the scheduling algorithm on 45nm CMOS ASIC has a clock period of 2.3ns, equivalent to 435MHz, for a 64-port OCS network. The high clock speed allows the scheduler to perform multiple iterations ($I = \lfloor E/\text{clk} \rfloor - b$) within an epoch (OCS switching rate). We have shown that the scheduler can configure the switch either every epoch (epoch-level) or every time-slot (slot-level). We have also shown that the timeslot level switching scheduling algorithm achieves a high throughput of 88-95% with a gain of 33-45% over its epoch-configured counterpart. A low average latency of less than 1.2μs is achieved by the slot-level scheduling algorithm compared to the 60-80μs latency incurred by epoch-level scheduling; this is proportional to the average transmitter buffer size (less than 12.8kB at operable load). The median and tail latency are also reduced by 2 orders of magnitude compared to epoch-level scheduling. Tuning at time-slot, rather than at epoch-level, increases wavelength usage to 100% from almost 60%. The size of the scheduler buffer is also as low as 1 MB for 2000 epochs, which is almost an order of magnitude lower in slot-level scheduling. The use of cascaded fast tunable DS-DBR lasers at the transmitter and fast coherent receivers for reception help to achieve a low network energy consumption of 82 pJ/bit and with 1.1-3.5 times cost reduction compared to an electronic network equivalent. The PULSE architecture scales to support up to 25.6 Pbps 4096-node network with 64 racks hosting 64 nodes each. A latency overhead of 0.15μs, 0.33μs and 1.12μs are shown for intra-rack, inter-rack and end-rack distances due to propagation delay.

Acknowledgment

This work is supported by EPSRC TRANSNET program (EP/R035342/1), by Microsoft Research through its PhD scholarship programme (T. Gerard) and the UCL-Cambridge CDT program in Integrated Photonic and Electronic Systems.

References

[1] Cisco. (2018) VNI: Forecast and Trends, 2017-2022. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[2] A. Singh *et al.*, "Jupiter Rising: A Decade of Clos Topologies and Centralized Control in Google's Datacenter Network," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, Aug. 2015.

[3] Y. Sverdlik. (2016) Intel: World Will Switch to "Scale" Data Centers by 2025. [Online]. Available: http://www.datacenterknowledge.com/archives/2016/04/22/intel-world-will-switch-to-scale-data-centers-by-2025

[4] T. Bawden. (2016) Global warming: Data centres to consume three times as much energy in next decade, experts warn. [Online]. Available: http://www.independent.co.uk/environment/global-warming-data-centres-to-consume-three-times-as-much-energy-in-next-decade-experts-warn-a6830086.html

[5] N. Rasmussen. (2009) Allocating Data Center Energy Costs and Carbon to IT Users. [Online]. Available: https://uat-s.insight.com/uk01/en-gb/content/media/apc-allocating-costs.pdf

[6] Cisco. (2018) 95% of Data Centre Traffic will come from Cloud by 2021. [Online]. Available: https://www.cloudpro.co.uk/leadership/7304/cisco-95-of-data-centre-traffic-will-come-from-cloud-by-2021

[7] K. Clark *et al.*, "Sub-Nanosecond Clock and Data Recovery in an Optically-Switched Data Centre Network," in *ECOC Post-deadline*, September 2018.

[8] Y. Xu *et al.*, "Bobtail - Avoiding Long Tails in the Cloud," in *10th USENIX Symposium on NSDI 13*, Lombard, IL, 2013, pp. 329–341.

[9] E. Agrell *et al.*, "Roadmap of optical communications," *Journal of Optics*, vol. 18, no. 6, p. 063002, 2016.

[10] H. J. Chao *et al.*, "Petastar: a petabit photonic packet switch," *IEEE J-SAC*, vol. 21, no. 7, pp. 1096–1112, Sept 2003.

[11] M. Duser and P. Bayvel, "Analysis of wavelength-routed optical burst-switched network performance," in *ECOC*, vol. 1, Sep. 2001, pp. 46–47.

[12] S. Di Lucente *et al.*, "Scaling low-latency optical packet switches to a thousand ports," *JOCN*, vol. 4, no. 9, pp. A17–A28, Sept 2012.

[13] B. R. Rofoee *et al.*, "Griffin: Programmable Optical DataCenter With SDN Enabled Function Planning and Virtualisation," *JLT*, vol. 33, no. 24, pp. 5164–5177, Dec 2015.

[14] Y. Cheng *et al.*, "POTORI: a passive optical ToR interconnect architecture for Data Centers," *JOCN*, vol. 9, no. 5, pp. 401–411, May 2017.

[15] R. Proietti *et al.*, "Scalable Optical Interconnect Architecture Using AWGR-Based TONAK LION Switch With Limited Number of Wavelengths," *JLT*, vol. 31, no. 24, pp. 4087–4097, Dec 2013.

[16] J. Gripp *et al.*, "IRIS optical packet router," *J. Opt. Netw.*, vol. 5, no. 8, pp. 589–597, Aug 2006.

[17] W. M. Mellette *et al.*, "A Scalable, Partially Configurable Optical Switch for Data Center Networks," *JLT*, vol. 35, no. 2, pp. 136–144, Jan 2017.

[18] N. Terzenidis *et al.*, "High-Port and Low-Latency Optical Switches for Disaggregated Data Centers: The Hipoλaos Switch Architecture," *JOCN*, vol. 10, no. 7, pp. 102–116, July 2018.

[19] F. Yan *et al.*, "Opsquare: A flat dcn architecture based on flow-controlled optical packet switches," *JOCN*, vol. 9, no. 4, pp. 291–303, April 2017.

[20] D. Alistarh *et al.*, "A High-Radix, Low-Latency Optical Switch for Data Centers," *SIGCOMM*, vol. 45, no. 4, pp. 367–368, Aug. 2015.

[21] A. Funnell *et al.*, "Hybrid Wavelength Switched-TDMA High Port Count All-Optical DC Switch," *JLT*, vol. 35, no. 20, pp. 4438–4444, Oct 2017.

[22] J. Benjamin *et al.*, "A High Speed Hardware Scheduler for 1000-Port OPSs to Enable Scalable Data Centers," in *HOTI*, Aug 2017, pp. 41–48.

[23] V. Natoli *et al.*, "A Decade of Accelerated Computing Augurs Well For GPUs," Jul 2019. [Online]. Available: https://www.nextplatform.com/2019/07/10/a-decade-of-accelerated-computing-augurs-well-for-gpus/

[24] T. Ray, "Seeking Big A.I. Advances, a Startup Turns to a Huge Computer Chip," Aug 2019. [Online]. Available: https://fortune-com.cdn.ampproject.org/c/s/fortune.com/2019/08/19/ai-artificial-intelligence-cerebras-wafer-scale-chip/amp/

[25] T. Baji, "Evolution of the GPU Device widely used in AI and Massive Parallel Processing," in *EDTM*, March 2018, pp. 7–9.

[26] "DARPA FastNICs Program Looks to Accelerate Application Performance by 100x," Sep 2019. [Online]. Available: https://insidehpc.com/2019/09/darpa-fastnics-program-looks-to-accelerate-application-performance-by-100x/

[27] N. Zilberman *et al.*, "Stardust: Divide and Conquer in the DCN," in *USENIX Symposium NSDI*, Boston, MA, 2019, pp. 141–160.

[28] A. Roy *et al.*, "Inside the Social Network's (Datacenter) Network," *SIGCOMM*, vol. 45, no. 4, pp. 123–137, Aug. 2015.

[29] A. Funnell *et al.*, "High port count hybrid wavelength switched TDMA optical switch for Data Centers," in *2016 OFC*, March 2016, pp. 1–3.

[30] P. Bayvel and M. Dueser, "Optical burst switching: research and applications," in *OFC*, vol. 2, Feb 2004, pp. 4 pp. vol.2–.

[31] R. Figueiredo *et al.*, "Hundred-Picoseconds Electro-Optical Switching With Semiconductor Optical Amplifiers Using Multi-Impulse Step Injection Current," *JLT*, vol. 33, no. 1, pp. 69–77, Jan 2015.

[32] "AN-2095 Controlling the S7500 CW Tunable Laser-RevA Updated," Sep 2019. [Online]. Available: https://www.finisar.com/sites/default/files/downloads/an-2095-controlling_the_s7500_cw_tunable_laser-reva_updated.pdf

[33] J. E. Simsarian, M. C. Larson, H. E. Garrett, Hong Xu, and T. A. Strand, "Less than 5-ns wavelength switching with an SG-DBR laser," *IEEE Photonics Technology Letters*, vol. 18, no. 4, pp. 565–567, Feb 2006.

[34] K. Szczerba *et al.*, "Energy Efficiency of VCSELs in the Context of Short-Range Optical Links," *IEEE Photonics Technology Letters*, vol. 27, no. 16, pp. 1749–1752, Aug 2015.

[35] A. Gaeta *et al.*, "Photonic-chip-based frequency combs," *Nature Photonics*, vol. 13, pp. 158–169, 03 2019.

[36] K. Morito, S. Tanaka, S. Tomabechi, and A. Kuramata, "A broad-band MQW semiconductor optical amplifier with high saturation output power and low noise figure," *IEEE Photonics Technology Letters*, vol. 17, no. 5, pp. 974–976, May 2005.

[37] M. Nagatani *et al.*, "A 3-Vppd 730-mW Linear Driver IC Using InP HBTs for Advanced Optical Modulations," *CSICS*, pp. 1–4, 2013.

[38] K. Grobe *et al.*, *Wavelength Division Multiplexing: A Practical Engineering Guide*, 1st ed. Wiley Publishing, 2013.

[39] P. Minzioni *et al.*, "Roadmap on all-optical processing," *Journal of Optics*, vol. 21, no. 6, 5 2019.

[40] T. Yoshimatsu *et al.*, "Compact and high-sensitivity 100-Gb/s (4× 25 Gb/s) APD-ROSA with a LAN-WDM PLC demultiplexer," *Opt. Express*, vol. 20, no. 26, pp. B393–B398, Dec 2012.

[41] G. Zervas *et al.*, "Optically Disaggregated Data Centers with minimal remote memory latency: Technologies, architectures, and resource allocation [Invited]," *JOCN*, vol. 10, no. 2, pp. A270–A285, Feb 2018.

[42] "Photonic ASIC directly integrates 100G optical ports," Jan 2019. [Online]. Available: https://www.eenewsanalog.com/news/photonic-asic-directly-integrates-100g-optical-ports

[43] R. Meade *et al.*, "TeraPHY: A High-Density Electronic-Photonic Chiplet for Optical I/O from a Multi-Chip Module," in *OFC*, 2019, pp. 1–3.

[44] T. Trader, "Ayar Labs to Demo Photonics Chiplet in FPGA Package at Hot Chips," Aug 2019. [Online]. Available: https://www.hpcwire.com/2019/08/19/ayar-labs-to-demo-photonics-chiplet-in-fpga-package-at-hot-chips/

[45] Finisar, "100GBASE-LR4 10 km Gen2 QSFP28 Optical Transceiver," Feb 2020. [Online]. Available: https://www.finisar.com/optical-transceivers/ftlc1154rdpl

[46] B. Lee, *Platforms for Integrated Photonic Switching Modules, OFC Workshop*, 2019.

[47] S. Lange, A. Raja, K. Shi, M. Karpov, R. Behrendt, D. Cletheroe, I. Haller, F. Karinou, X. Fu, J. Liu, A. . Lukashchuk, B. Thomsen, K. Jozwik, P. Costa, T. J. Kippenberg, and H. Ballani, "Sub-nanosecond optical switching using chip-based soliton microcombs," in *Optical Fiber Communication Conference (OFC'20)*. The Optical Society (OSA), March 2020. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sub-nanosecond-optical-switching-using-chip-based-soliton-microcombs/

[48] *High Speed Ethernet Optics Report*, 2018. [Online]. Available: https://www.lightcounting.com/DataCenter.cfm

[49] "Arista Networks 7260CX3-64 Layer 3 Switch," Sep 2019. [Online]. Available: https://www.pcnation.com/web/details/4AP124/arista-networks-7260cx3-64-layer-3-switch-dcs-7260cx3-64-r

[50] "Arista Networks 7170-64C Layer 3 Switch," Sep 2019. [Online]. Available: https://www.neophotonics.com/800g-coherent-versus-pam4-optical-transceivers-data-centers/

[51] G. Arevalo *et al.*, "Optimization of multiple PON deployment costs and comparison between GPON, XGPON, NGPON2 and UDWDM PON," *OSN*, vol. 25, 03 2017.

[52] "Generic Compatible 100GBASE-SR4 QSFP28 850nm 100m DOM Transceiver Module," Sep 2019. [Online]. Available: https://www.fs.com/products/75308.html

[53] T. Theocharidis, "Public executive summary of the final Project Periodic Report," Mar 2016. [Online]. Available: https://cordis.europa.eu/docs/projects/cnect/4/318714/080/reports/001-ASTRONPublicexecutivesummaryofthefinalProjectPeriodicReport.pdf

[54] Z. Jia *et al.*, *Digital Coherent Transmission for Next Generation Cable Operators' Optical Access Networks*, 2017. [Online]. Available: https://www.nctatechnicalpapers.com/Paper/2017/2017-digital-coherent-transmission-for-nextgeneration-cable-operators-optical-access-networks

[55] "800G: Coherent versus PAM4 Optical Transceivers Inside Data Centers," Sep 2019. [Online]. Available: https://www.neophotonics.com/800g-coherent-versus-pam4-optical-transceivers-data-centers/

[56] N. Farrington *et al.*, "HELIOS: A Hybrid Electrical/Optical Switch Architecture for Modular DCNs," *SIGCOMM*, vol. 41, no. 4, Aug. 2010.

[57] K. Chen *et al.*, "OSA: An Optical Switching Architecture for Data Center Networks With Unprecedented Flexibility," *IEEE/ACM TON*, vol. 22, no. 2, pp. 498–511, April 2014.

[58] W. Mellette *et al.*, "RotorNet: A Scalable, Low-complexity, Optical Datacenter Network," *SIGCOMM*, pp. 267–280, 2017.

[59] N. Hamedazimi *et al.*, "FireFly: A Reconfigurable Wireless Data Center Fabric Using Free-space Optics," *SIGCOMM*, vol. 44, no. 4, pp. 319–330, Aug. 2014.

[60] H. Liu *et al.*, "Circuit Switching Under the Radar with REACToR," in *NSDI*. Berkeley, CA, USA: USENIX Association, 2014, pp. 1–15.

[61] N. Farrington *et al.*, "A Multiport Microsecond Optical Circuit Switch for Data Center Networking," *IEEE Photonics Technology Letters*, vol. 25, no. 16, pp. 1589–1592, Aug 2013.

[62] H. Chao, K. Deng, and Z. Jing, "A petabit photonic packet switch (p3s)," in *Proceedings - IEEE INFOCOM*, vol. 1, 2003, pp. 775–785.

[63] X. Ye *et al.*, "DOS - A Scalable Optical Switch for Data Centers," in *ACM/IEEE ANCS*, Oct 2010, pp. 1–12.

[64] D. Hartman *et al.*, "Optical clock distribution using a mode-locked semiconductor laser diode system," 1991, p. FC3.

[65] M. Inggs *et al.*, "Investigation of white rabbit for synchronization and timing of netted radar," in *Radar Conference*, Oct 2015, pp. 214–217.