Software-defined networking control plane for seamless integration of multiple silicon photonic switches in Datacom networks

YIWEN SHEN, MAARTEN H. N. HATTINK, PAYMAN SAMADI, QIXIANG CHENG, ZIYI ZHU, ALEXANDER GAZMAN, AND KEREN BERGMAN

Lightwave Research Laboratories, Electrical Engineering Department, Columbia University, U.S.A.
ys2799@columbia.edu

Abstract: Silicon photonics based switches offer an effective option for the delivery of dynamic bandwidth for future large-scale Datacom systems while maintaining scalable energy efficiency. The integration of a silicon photonics-based optical switching fabric within electronic Datacom architectures requires novel network topologies and arbitration strategies to effectively manage the active elements in the network. We present a scalable software-defined networking control plane to integrate silicon photonic based switches with conventional Ethernet or InfiniBand networks. Our software-defined control plane manages both electronic packet switches and multiple silicon photonic switches for simultaneous packet and circuit switching. We built an experimental Dragonfly network testbed with 16 electronic packet switches and 2 silicon photonic switches to evaluate our control plane. Observed latencies occupied by each step of the switching procedure demonstrate a total of 344 µs control plane latency for data-center and high performance computing platforms.

OCIS codes: (060.4253) Networks, circuit-switched; (060.4510) Optical communications; (200.4650) Optical interconnects.

References and links

1. K. Wen, P. Samadi, S. Rumley, C. Chen, Y. Shen, M. Bahadori, K. Bergman, and J. Wilke, "Flexfly: enabling a reconfigurable Dragonfly through silicon hotonics,", in Proceedings of SC16: International Conference for High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, 2016, pp. 166–177.
2. K. Wen, D. Calhoun, S. Rumley, X. Zhu, Y. Liu, L.W. Luo, R. Ding, T.B. Jones, M. Hochberg, and M. Lipson, "Reuse distance based circuit replacement in silicon photonic interconnection networks for HPC," in Proceedings of IEEE 22nd Annual Symposium on High-Performance Interconnects, Mountain View, CA, (IEEE, 2014) pp. 49–56.
3. B. Robertson, H. Yang, M. M. Redmond, N. Collings, J. R. Moore, J. Liu, A. M. Jeziorska-Chapman, M. Pivnenko, S. Lee, A. Wonfor, I. H. White, W. A. Crossland, and D. P. Chu, "Demonstration of multi-casting in a 1 × 9 LCOS wavelength selective switch," J. Lightwave. Technol. **32**(3), 402–410 (2014).
4. A. S. Hamza, J. S. Deogun, and D. R. Alexander, "Free space optical multicast crossbar," in IEEE/OSA Journal of Optical Communications and Networking, vol. **8**, no. 1, pp. 1–10, January 1 2016.
5. H. Chen, N. K. Fontaine, B. Huang, X. Xiao, R. Ryf and D. T. Neilson, "Wavelength Selective Switch for Dynamic VCSEL-Based Data Centers," presented at ECOC 2016 - Post Deadline Paper; 42nd European Conference on Optical Communication, Dusseldorf, Germany, pp. 1–3. 2016.
6. I. Stanimirovic, and Z. Stanimirovic, "Optical MEMS for telecommunications: some reliability issues," Advances in Micro/Nano Electromechanical Systems and Fabrication Technologies (2013), Available from: https://www.intechopen.com/books/advances-in-micro-nano-electromechanical-systems-and-fabrication-technologies/optical-mems-for-telecommunications-some-reliability-issues
7. T. Komljenovic, M. Davenport, J. Hulme, A. Y. Liu, C. T. Santis, A. Spott, S. Srinivasan, E. J. Stanton, C. Zhang, and J. E. Bowers, "Heterogeneous silicon photonic integrated circuits," in Journal of Lightwave Technology, vol. **34**, no. 1, pp. 20–35, Jan 1, 2016.
8. L. Alloatti, M. Wade, V. Stojanovic, M. Popovic, and R. J. Ram, "Photonics design tool for advanced CMOS nodes," IET Optoelectronics **9**(4), 163–167 (2015).
9. Q. Cheng, M. Bahadori, S. Rumley, and K. Bergman, "Highly-scalable, low-crosstalk architecture for ring-based optical space switch fabrics," in Proceedings of IEEE Optical Interconnects Conference (IEEE, 2017) pp. 41–42.
10. K. Tanizawa, K. Suzuki, M. Toyama, M. Ohtsuka, N. Yokoyama, K. Matsumaro, M. Seki, K. Koshino, T. Sugaya, S. Suda, G. Cong, T. Kimura, K. Ikeda, S. Namiki, and H. Kawashima, "Ultra-compact 32 × 32 strictly-non-blocking Si-wire optical switch with fan-out LGA interposer," Opt. Express **23**, 17599–17606 (2015).

11. L. Lu, S. Zhao, L. Zhou, D. Li, Z. Li, M. Wang, X. Li, and J. Chen, "16 × 16 non-blocking silicon optical switch based on electro-optic Mach-Zehnder interferometers," Opt. Express **24**, 9295–9307 (2016).
12. L. Qiao, W. Tang, and T. Chu, "32 × 32 silicon electro-optic switch with built-in monitors and balanced-status units," Scientific Reports **7**, 2017.
13. T. J. Seok, N. Quack, S. Han, R. S. Muller, and M. C. Wu, "Large-scale broadband digital silicon photonic switches with vertical adiabatic couplers," Optica **3**, 64–70 (2016).
14. D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-defined networking: a comprehensive survey," in Proceedings of the IEEE **103** (1), pp. 14–76, Jan. 2015.
15. Y. Shen, P. Samadi, Z. Zhu, A. Gazman, E. Anderson, D. Calhoun, M. Hattink, and K. Bergman, "Software-defined networking control plane for seamless integration of silicon photonics in Datacom networks," presented at ECOC 2017, 43rd European Conference and Exhibition on Optical Communications, Gothenburg, Sweden, 2017.
16. Ryu SDN Controller, URL: "https://osrg.github.io/ryu/"
17. J. Kim, W. J. Dally, S. Scott and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in Proceedings of International Symposium on Computer Architecture, Beijing, 2008, pp. 77–88.
18. D. Nikolova, S. Rumley, D. Calhoun, Q. Li, R. Hendry, P. Samadi, and K. Bergman, "Scaling silicon photonic switch fabrics for data center interconnection networks," Opt. Exp. **23**, pp. 1159–1175 (2015).
19. K. Ishii, T. Inoue, and S. Namiki, "Toward exa-scale optical circuit switch interconnect networks for future datacenter/HPC," Proc. SPIE **10131** (2017).
20. T. Shiraishi, Q. Li, Y. Liu, X. Zhu, K. Padmaraju, R. Ding, M. Hochberg, and K. Bergman, "A reconfigurable and redundant optically-connected memory system using a silicon photonic switch," in Optical Fiber Communication Conference (Optical Society of America, 2014), paper Th2A.10.
21. R. Aguinaldo, A. Forencich, C. DeRose, A. Lentine, D. Trotter, Y. Fainman, G. Porter, G. Papen, and S. Mookherjea, "Wideband silicon-photonic thermo-optic switch in a wavelength-division multiplexed ring network," Opt. Exp **22**, pp.8205–8218 (2014).
22. A. Gazman, C. Browning, M. Bahadori, Z. Zhu, P. Samadi, S. Rumley, V. Vujicic, L. Barry, and K. Bergman, "Software-defined control-plane for wavelength selective unicast and multicast of optical data in a silicon photonic platform," Opt. Exp **25**, pp. 232–242 (2017).
23. L. Liu, R. Munoz, R. Casellas, T. Tsuritani, R. Martinez and I. Morita, "OpenSlice: An OpenFlow-based control plane for spectrum sliced elastic optical path networks," Opt. Express **21**, 4194–4204 (2013).
24. Z. Zhu, C. Chen, X. Chen, S. Ma, L. Liu, X. Feng, and S. J. B. Yoo, "Demonstration of cooperative resource allocation in an OpenFlow-controlled multidomain and multinational SD-EON testbed," in Journal of Lightwave Technology, vol. **33**, no. 8, pp. 1508–1514, April 15, 2015.
25. R. Casellas, R. Martinez, R. Munoz, R. Vilalta, L. Liu, T. Tsuritani, and I. Morita, "Control and management of flexi-grid optical networks with an integrated stateful path computation element and OpenFlow controller [invited]," in IEEE/OSA Journal of Optical Communications and Networking, vol. **5**, no. 10, pp. A57–A65, Oct. 2013.
26. J. Yin, J. Guo, B. Kong, H. Yin, and Z. Zhu, "Experimental demonstration of building and operating QoS-aware survivable vSD-EONs with transparent resiliency," Opt. Express **25**, 15468–15480 (2017).
27. Wireshark, URL: https://www.wireshark.org/
28. FlowMod, URL: http://flowgrammable.org/sdn/openflow/messagelayer/flowmod/#tab_ofp_1_3_1
29. Y. Huang, Q. Cheng, N. C. Abrams, J. Zhou, S. Rumley, and K. Bergman, "Automated calibration and characterization for scalable integrated optical switch fabrics without built-in power monitors", presented at ECOC 2017, 43rd European Conference and Exhibition on Optical Communications, Gothenburg, Sweden, 2017.
30. Q. Cheng, M. Bahadori, Y. Huang, S. Rumley, and K. Bergman, "Smart routing tables for integrated photonic switch fabrics,", presented at ECOC 2017, 43rd European Conference and Exhibition on Optical Communications, Gothenburg, Sweden, 2017.
31. tcpdump, URL: https://www.tcpdump.org
32. iPerf, "iPerf - The ultimate speed test tool for TCP, UDP and SCTP", URL: https://iperf.fr/
33. Interconnect Analysis: 10GigE and InfiniBand in high performance computing, White Paper, (HPC AI Advisory Council, 2009). http://www.hpcadvisorycouncil.com/pdf/IB_and_10GigE_in_HPC.pdf
34. Y. Shen, A. Gazman, Z. Zhu, M. Y. Teh, M. Hattink, S. Rumley, P. Samadi, and K. Bergman, "Autonomous dynamic bandwidth steering with silicon photonic-based wavelength and spatial switching for Datacom networks," in Optical Fiber Communication Conference, OSA Technical Digest (online) (Optical Society of America, 2018), paper Tu3F.2.
35. A. Rylyakof, J. Proesel, S. Rylov, B. Lee, J. Bulzhacchelli, A. Ardey, B. Parker, M. Beakes, C. Baks, C. Schow, and M. Meghelli, "22.1 A 25Gb/s burst-mode receiver for rapidly reconfigurable optical networks," in Proceedings of International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers, (IEEE, 2015) pp. 1–3.
36. Q. Cheng, M. Bahadori, and K. Bergman, "Advanced path mapping for silicon photonic switch fabrics" in Conference on Lasers and Electro-Optics, OSA Technical Digest (online) (Optical Society of America, 2017), paper SW1O.5.

1. Introduction

The increasing growth of both data center (DC) and high performance computing (HPC) applications has created a dependency on the ability of the interconnection network to provide

greater bandwidth capacities than ever before. As these network infrastructures grow in scale, they are also expected to become more efficient in their power consumption and equipment costs. Current static, best-for-all network topologies are inadequate to meet these new requirements. This is because DC and HPC applications not only require the movement of increasingly large quantities of data, each application also features unique traffic characteristics and latency requirements between neighbouring nodes. These challenges lead to over-provisioned static links, which are expensive and inefficient. Furthermore, many HPC applications have traffic characteristics that vary slowly over the runtime of the application, and traffic only occurs between a small number of source-destination pairs [1]. The use of a static network for HPC applications equates to a mismatch between the network topology and the application's traffic pattern, and results in data-starved processors connected by oversubscribed links, as well as wasted bandwidth allocated on links with minimal or no traffic.

These problems can be resolved through the use of flexible networks that introduce new degrees of freedom by enabling physical layer reconfiguration for cluster-to-cluster communication. With an agile optical switching fabric, the network can rapidly allocate extra bandwidth to traffic intensive source-destination pairs from the links that do not send traffic, thereby significantly improving performance through efficient network resource utilization [2]. This rapid network resource allocation can be achieved with optical switches that manipulate the path taken by the optical signal by rewiring the connections between electronic endpoints, which can be both compute nodes or electronic routers.

Current forms of optical switching rely on discrete optical elements such as micro electro-mechanical systems (MEMS) actuated micro-mirror arrays or liquid crystal on silicon (LCOS) matrices [3–5]. These systems can be expensive, bulky, and their mechanical elements are susceptible to failure from a variety of environmental factors [6]. Over the last decade, silicon photonics (SiP) based switches have emerged as a promising technology for realizing fast and high bandwidth switching with minimal energy consumption. Because silicon photonics are fabricated under highly integrated platforms using manufacturing processes compatible with CMOS devices, they have the advantages of small area footprint, low power consumption, low fabrication costs at large scales, and potential for nanosecond range dynamic connectivity [7–9]. In recent years, state of the art SiP switches with up to 32×32 ports have been demonstrated with MZI elements [10–12], and a 64×64 port SiP switch with MEMS-actuated adiabatic couplers [13].

By integrating silicon photonics based interconnection networks into our existing DC and HPC infrastructures, we can build network architectures and topologies that dramatically improve conventional electronics based architectures with the unique characteristics of an optical switch fabric. Controlling the switching modality of the electronic and optical switching elements in the network in a synchronized manner can be achieved through Software-Defined Networking (SDN) [14]. SDN refers to the decoupling of the control plane from the data plane, along with the use of a centralized controller to provide routing and forwarding decisions based on a global view of network traffic behavior. Specifically, an SDN application that operates on top of the SDN controller framework provides advanced network reconfiguration capabilities by modifying the flow tables in the electronic switches. Additional modifications to the SDN application and supplemental hardware also enable the management of the silicon photonic switches. A unified control plane that manages both types of switches together allows for synchronicity during switching operations and thus minimizes switching latency.

In this work, we investigate the feasibility and performance of integrating silicon photonic circuit switching in conventional electronic Datacom network architectures, extending from [15]. Our SDN control plane architecture seamlessly integrates Mach-Zehnder Interferometer (MZI) based SiP switches with commercial electronic top-of-rack (ToR) packet switches. It is a unified control plane written within the Ryu SDN controller framework [16], designed to scale in order

to manage a data plane consisting of many SiP and electronic packet switches (EPSs). This is enabled by the SDN controller's capability to manage the EPSs and SiP switches with parallel commands over Ethernet. The network architecture is optimized and synchronized with Ethernet for minimal packet drop, and its architectural concept is adaptable to any other standard SDN controller as well as HPC InfiniBand networks. We fully implemented the control plane and built a Dragonfly [17] network testbed for evaluations, and demonstrate simultaneous control of two SiP switches for dynamic bandwidth steering. Our experiments demonstrate network optimization under bulk data transfers. We evaluate the latency of the control plane and the SiP and electronic switching elements to provide a full overview of the latencies at every layer of the network architecture. Experimental results show end-to-end control plane latency of 344 µs.

In section 2, we review related works on silicon photonics switching fabrics in electronic networks. In section 3, we describe the control plane, network architecture, control messaging protocol, and system hardware. Section 4 presents the testbed we built to evaluate our control plane and section 5 presents the experimental results. In section 6 we discuss the applications, scalability, flexibility, and obstacles faced by our proposed architecture. Lastly, section 7 presents our conclusion.

2. Related works

There has been a number of works relating to the integration of SiP switching into DC and HPC networks, as well as software-defined control planes for reconfigurable optical networks. With regards to the works involving SiP switches, many of them can be categorized into either theoretical works that consider large scale SiP based network architectures [18, 19], or experimental works that focuses on the operation and control of the SiP element itself with less emphasis on the system [20–22].

Theoretical control planes integrating SiP switching include [18] which analyzes the feasibility of using a theoretical high radix 128×128 micro-ring based single chip switching fabric in DC networks, by considering the device's total and crosstalk power penalties, as well as its control mechanism by field programmable gate arrays (FPGAs). [19] discusses a SiP based circuit switching interconnect architecture that is able to scale of more than 100,000 endpoints through wavelength division multiplexing (WDM) assignments. On the experimental side, there have been system level demonstrations of both MZI based [20, 21] and micro-ring based SiP switches [22]. [20] demonstrates 10 Gb/s WDM communication between a CPU and two memory nodes that are emulated by FPGAs. Operations within a larger system are demonstrated in [21], where an MZI based SiP device switched twenty 10 Gb/s wavelengths simultaneously in an optical ring network. [22] demonstrates an SiP 1×8 micro-ring array chip in conjunction with a fast tunable laser capable of perfoming unicast and multicast, and is capable of being controlled by user input through an FPGA interface. In these experimental works [20–22], the control plane was focused on controlling the specific parameters of the device at the physical level. There is little or no mention of methods to integrate the device in the network layer under a real DC or HPC setting, and there are only a few evaluations of the device's performance or scalability under large network environments. For the theoretical works [18, 19] that do tackle the feasibility of using SiP switching under such environments, they cannot expect and/or evaluate the nitty-gritty details involved in the synchronization between the SiP elements and the conventional hardware in electronic networks. Therefore, the middle ground between these two areas, i.e., the exploration of integrating SiP switching elements into a real DC or HPC environment, using an SDN control plane that is designed to arbitrate the switching operations between multiple server clusters using SiP switches, and analyzing its performance in terms of control plane protocol, latency and scalability is what this work seeks to address.

There have also been many efforts in software-defined elastic optical networks (SD-EONs) which feature a similar control plane architecture to this work, but integrates bandwidth variable

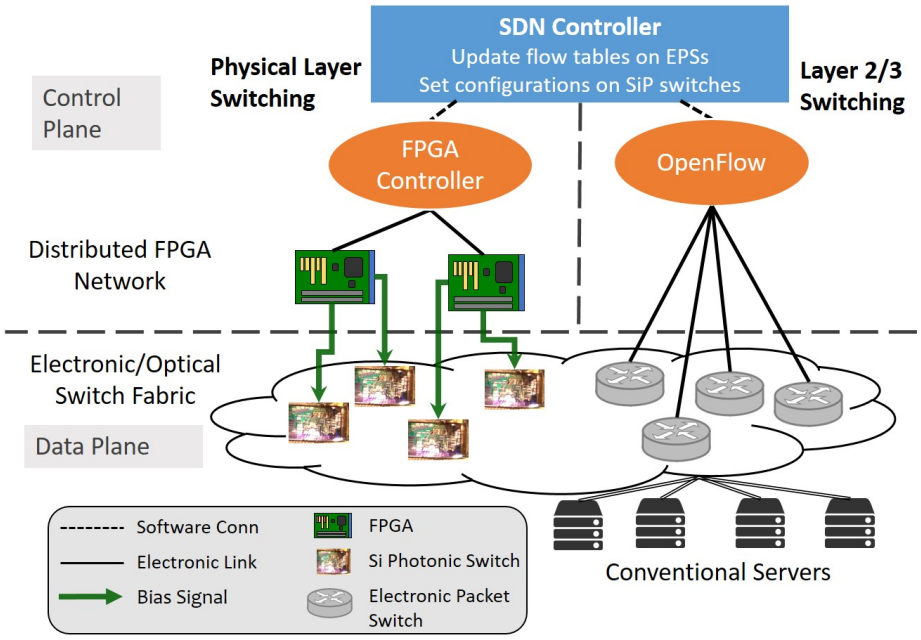


Fig. 1. Network architecture divided into the control plane and data plane, and consists of unified SDN management of SiP physical layer switching and electronic L2/3 switching

wavelength selective switches (BV-WSSs) instead of SiP switches in the data plane to allocate variable spectrums for dynamic optical path provisioning [23–26]. For example, [23] demonstrates a control plane that features a protocol converter extended from the OpenFlow protocol for processing incoming packet requests, a cross-connection table for maintaining connection information, and the ability to control emulated BV-WSSs to allocate spectrums for varying data rates ranging from 100 Gbps to 400 Gbps connections as well as different modulation formats. Similarly, [24] showed SD-EON over multi-domains, using OpenFlow to interface with BV-WSSs and featuring a path computation element (PCE) for service provisioning, with 272 ms control plane latency. [25] combines a path computation element (PCE) with OpenFlow controller both externally and as a fully integrated model, to configure emulated flex-grid optical hardware with a latency of approximately 50 ms. Lastly, [26] further enhances SD-EONS with virtual SD-EONs to show autonomous quality of service (QoS)-aware lightpath activation and restoration. While the control planes for using BV-WSSs to allocate wavelength/spectrum are well-developed, most research efforts involving SiP switches still remain at the physical layer, and the arbitration strategies required to utilize SiP switches in a large scale experimental Datacom testbed has been a relatively unexplored area. We attempt to address this by developing a scalable SDN control plane to utilize SiP switches to perform optical circuit switch based bandwidth steering. We also evaluate our control plane by measuring each of the latencies associated with every step of both the software and hardware layers involved on a physical network testbed. In our current control plane, the OpenFlow protocol is only used for managing the electronic packet switches, while the SiP switches are managed using an in-house protocol, described in the next section in more detail.

3. Network architecture

Our proposed network architecture is shown in Fig. 1. At the top is the centralized SDN controller that manages the behavior of all the switching elements in the network, which will be discussed

in detail below. From there, the network architecture is divided horizontally into the control plane and data plane layers, and vertically into the physical layer switching and layer 2/3 switching sections. The data plane layer consists of the electronic and optical switch fabric which are composed of the silicon photonic elements and electronic packet switches (EPSs). Servers are connected to the EPSs first, while the SiP switches are placed in-between EPS connections to serve as dynamic inter-group connections. The SDN controller is designated to be the central point of management to reduce the delay between the switching modality of the electronic packet switches and SiP switches. This arises from the fact that both types of switches must work together in tandem to create an end-to-end path for the packets to travel between servers. Further details are below.

3.1. Layer 2/3 (electronic packet) switching

The electronic packet switches (EPSs) dictates the path taken by TCP/IP packets in between the servers. Their behavior is managed by the SDN controller through the OpenFlow protocol, which is a standard southbound application programming interface (API) protocol used commonly by SDN applications. In our work, the SDN application plays the role of adding or deleting layer 3 flow rules to the flow tables of the EPSs. The flow rules consist of many fields, but for our purposes the most relevant fields are the in-port and out-port values, as well as the IP destination address. These three fields determine the route that is taken by the packets from the source node to the destination node. Since the SiP switches modify the physical topology of the network and connects different endpoints together that change over time, the SDN controller must be able to add/delete flows to coordinate the correct in-port and out-port values as well as the IP destination address with the topological change caused by the SiP switch, so that packets can flow smoothly from the source to the destination servers. Otherwise the packets will be dropped by the EPS connected to the destination compute node.

Each source-destination node pair that is the minimum distance (1 hop) from each other requires 4 individual flows in the flow table - the IP flow, ARP flow, and the same flows but with the in-port and out-port flipped and a different IP destination for the reverse direction as to make the connection bidirectional. These flow rules on the EPS allow the switch to forward IP and ARP traffic, both of which are required for two servers to communicate with each other. The first packet sent by a source server is the ARP request to learn the MAC address of the server corresponding to the destination IP address, and only then can the two servers exchange IP data packets. A hop refers to the connection between EPSs, which may or may not span across the SiP switch. Each additional hop to another EPS in the path requires 4 additional flows.

The number of bytes required to send a single flow modification to the EPS averages around 150 bytes, which we found using Wireshark [27]. A flow table modification is performed by sending the FlowMod message type from the SDN controller to the EPS, whose frame structure is outlined in [28] . This includes the header (4 bytes), and various key fields including the flow priority (2 bytes), out_port (4 bytes), match (4 bytes), and instruction (4 bytes), among others.

3.2. Physical layer (silicon photonic) switching

In order for the SDN application to control the SiP switches, a distributed FPGA network is required. This network is a 1G out-of-band Ethernet network that can scale in a fat-tree topology.

Each FPGA unit is equipped with Digital-to-Analog Converter (DAC) chips, which allows per-defined voltages/currents to be applied to the SiP switches. For the SDN application to control the FPGA network, an in-house C++ application was developed, called the FPGA Controller. The FPGA Controller acts as an intermediary to allow the SDN controller to communicate with the FPGA microcontroller. To do this, the SDN controller application first connects to the FPGA Controller with a TCP socket connection. During a physical topology change, the SDN application simultaneously modifies both the flow rules on the EPS and sends a flow mod

message to the FPGA Controller to set a given SiP switch to a specific configuration by sending which input and output port numbers of the SiP are to be connected. When the FPGA Controller application obtains the message from the SDN controller, it prepares a flow update Ethernet packet that contains the input and output port numbers to the FPGA unit that controls the relevant SiP element. The format of this packet is described in Section 3.2.1. Once the packet is received by the FPGA microcontroller, the device port numbers that are to be connected are read and mapped to a set of pre-defined voltages hardcoded into individual registers of the FPGA. These voltage values are then applied to the DACs of the FPGA in order to bias the SiP switch to change its configuration. A 5× amplification is required to deliver enough power the cause the SiP element to perform the state change. This amplification is performed by a DAC gain stage implemented on a printed circuit board (PCB). The amplified bias signals maintain their voltage levels until a new state change command from the SDN controller has been received. More details about procedures performed by the FPGAs are provided in Section 3.2.2 FPGA Operation Procedure.

While the in-house FPGA Controller software serves the purposes to communicate with the FPGA network to control SiP switches in this work, in future works an OpenFlow client for the FPGAs will be developed so that the control plane unifies its southbound APIs to the EPSs and the SiP switches to improve compatibility and scalability. This will also enable further reductions in latency as the SDN controller will be able to directly communicate with the FPGA without an intermediary module. Additional performance improvements can be done by integrating ARM processors to run the OpenFlow client and use hardware to decode extracted flow updates and offloading the tasks needed to be performed by the FPGA CPU. This can raise the performance and reduce latency in processing flow modification commands to what is achievable by application-specific integrated circuits (ASICs) without significantly raising costs or limiting reconfigurability options.

3.2.1. SiP switch control messaging protocol

A packet protocol under the Ethernet II frame format was developed to provide communication between the SDN controller and the FPGA, or between FPGA nodes. This common frame format allows for ease of use and scalability, as additional FPGAs can be added to the system simply by connecting it to the commercial Ethernet switch that connects all the FPGAs to the SDN controller server.

The format of the custom Ethernet frames is presented below in Table 1 and 2. Two types of custom protocols are used: the FPGA Ethernet Protocol and the Register Ethernet Protocol. The FPGA Ethernet Protocol is for communication between nodes (server to FPGA, or between FPGAs), and contains rudimentary features to handle packet loss using the Message ID field to allow for retransmission. The Register Ethernet Protocol is used during reading to or writing from the FPGA's memory-mapped registers. The data from the Register Ethernet Protocol is placed in the payload section of the FPGA Ethernet packet. Inside the Register Ethernet Protocol, the payload section contains two integers, each contained within 8 bits, that represent the input and output port values of the SiP switch that are to be connected.

The number of bytes used to communicate between the FPGA Controller and FPGAs using the FPGA Ethernet Protocol has a minimum size of 60 bytes (for a single pair of input and output ports), not counting the frame check sequence (FCS) that is 4 bytes long. 8 bytes are needed for every additional pair of input and output ports in the same packet.

3.2.2. FPGA operation procedure

The procedures performed by the FPGA after receiving a flow update packet are depicted in Fig. 2. During the initialization phase, the voltage values associated with each possible configuration of the SiP switch is hardcoded into the FPGA's registers. Once the FPGA receives flow update

Tab 1. FPGA Ethernet Protocol

Byte Position	Bit Position	Field Name	Description
0 to 13		Ethernet header	Source MAC Address and the Destination MAC Address, plus the EtherType.
14	7	REQ/ACK Bit	Indicates if a message is a request or response (=0) or an acknowledgement to such a request or response (=1).
14	6 to 0	Message Type	Indicates the type of protocol that send the packet. The Register Ethernet protocol uses the value 0.
15 to 17		Message ID	The sequence number of the message. Both communicating nodes keep a counter for both the other's message ID and their own. Each new transaction raises the counter by 2.
18 to 1514 (max)		Payload	

Tab 2. Register Ethernet Protocol

Byte Position	Bit Position	Field Name	Description
0 to 1		Address	Address being written to or read from. The register interface is 32 bit addressed.
2 to 3	0 to 13	Count	Number of 32 bits being read or written or sent as a response in this command.
3	6 to 0	Response Bit	1 if the command is a response type, else 0.
3	7	Write Bit	1 if command is a write type, else 0.
4 to 4+(4*Count-1)		Payload	In case of a WRITE or RESPONSE command the 4 byte command is followed by the data.

packets from the FPGA Controller, it parses the port numbers that are to be connected and uses it to determine the new switch configuration, which are saved using flip-flops so that it functions as a Moore finite state machine (FSM). From there, the switch configuration is mapped to the addresses of the registers that contain the voltage values associated with the new switch configuration. Therefore, the FSM is used to select the corresponding registers whose voltage values are then read in parallel and applied to the DACs. DAC voltage values can be read in parallel because separate memory registers are used to hold the voltage values (see Fig. 6). In the diagram shown above, two registers containing the hardcoded voltage values are assigned to each, but in general multiple registers containing multiple voltage values can be assigned to each DAC for larger SiP devices with a greater number of possible configurations. In fact, a single FPGA can be used to house hundreds of configurations given the size of its memory blocks used to hold voltage values in the registers. For example, a typical FPGA holds hundreds to thousands of memory blocks, each of which has tens of kilobits per block, which can be configured to have various widths and depths. If we configure an Altera M20K block which has 20×1024 bits into blocks that are 36 wide and 512 deep, and using 12-bit DACs, samples for 3 DACs and 512 unique configurations can be stored per memory block. With plentiful amount of memory available to store voltage values, we can be assured that voltage values can be stored in separate registers in the memory block which can be read in parallel, so that as the network scales up and more SiP devices are added, the latency for switching will not increase as more SiP elements are required to change configurations.

3.2.3. Control hardware scalability

The amount of control hardware required for scaling with larger SiP devices is related to the number of switching elements of the device, which is dependent on the driving scheme, the switch architecture and the port count of the device. Generally, there are two driving schemes

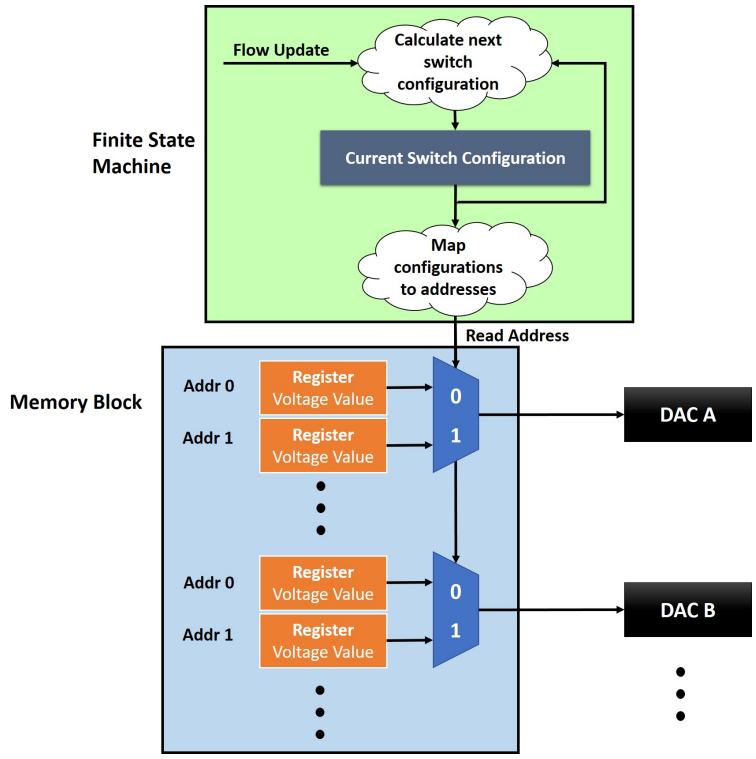


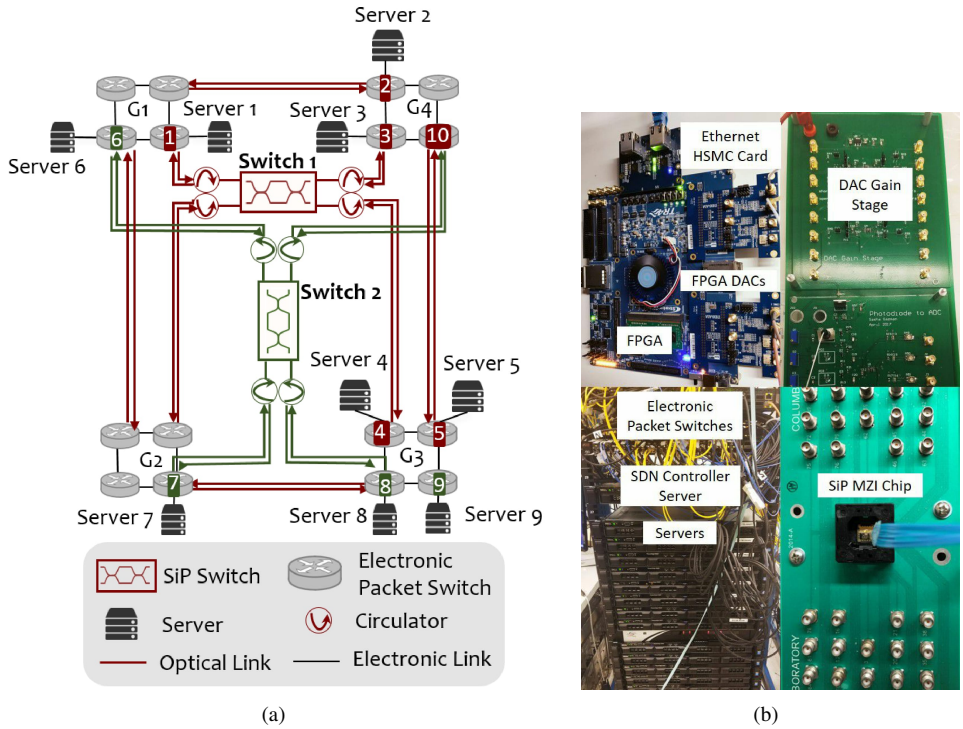
Fig. 2. The FPGA contains an FSM to process the incoming flow update packets and map them to registers in the memory block. These registers contains the voltage values that are applied to the DACs to configure the SiP switch elements

for electro-optic Mach-Zehnder type switches - single-armed and push-pull control [29]. Single-armed bias is where only one electro-optic phase modulator of the MZI element is used for switching, and this often requires an additional thermal-optic phase shifter for calibration so that the MZI element is in the cross/bar configuration. In the single-arm drive case, the bias voltage will be V_π to switch to the opposite state. This method requires higher drive voltage and induces higher electro-absorption loss, but it is favorable as it only requires one DAC to control an MZI element. On the other hand, the push-pull scheme requires to control both arms and thus requires two DACs to control one switch element. The driving voltage required is below V_π. In this work, we applied the single-arm drive scheme to take the advantage of using smaller number of DACs.

The architecture of the device determines the number of switching elements. Commonly applied architectures for optical switches include crossbar, Benes, dilated Benes, Banyan, N-stage planar, etc. The Benes architecture, which we chose in our experiments shown in Section 4, requires the least number of switching elements to achieve non-blocking connections for an $N \times N$ switch [30]. The $N \times N$ Benes switch has a total number of switching elements as $\frac{N}{2}(2\log_2 N - 1)$. Together with the single-arm driving scheme, we can therefore determine the number of DACs to control an $N \times N$ switch scales $\frac{N}{2}(2\log_2 N - 1)$.

4. Testbed

We built a testbed network to demonstrate the feasibility of integrating multiple SiP switches into an electronic network and to evaluate its performance metrics, shown in Fig. 3(a). It has a DragonFly topology, consisting of 4 groups (G1 to G4), with 4 EPSs (EPS1 to EPS4) in each



(a) (b)

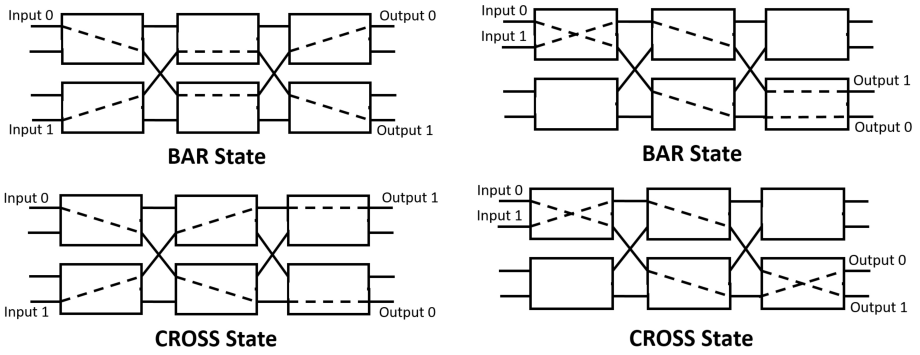
Fig. 3. (a) Topology of the experimental testbed, consisting of a Dragonfly network and 2 MZI switches of inter-group reconfiguration (b) Snapshot of the testbed

group. Each EPS has 2 servers connected to them. In Fig. 3(a) we label only the servers and EPSs that are relevant to our experiments, shown in the next section. The intra-group connections are 10G Direct-Attached copper cables, while the inter-group connections are optical links with 10G SFP+ transceivers with 24 dB power budget transmitting wavelengths in the ranges from C28 to C38 (1554.94 nm to 1546.92 nm). The EPSs are bridges created on 2 Pica8 Ethernet OpenFlow enabled switches, and they are connected to a separate SDN controller server via 1G campus internet, running a Ryu SDN application. The servers are equipped with a 10G Network Interface Card (NIC), an Intel Xeon 6-core processor and 24 GB of RAM, running Scientific Linux 7 (CentOS 7).

There are two SiP switches that connects the four groups of the DragonFly through the EPSs with circulators between each connection. In their default bar state, Switch 1 provides improved connectivity between the left and right halves of the network, while Switch 2 provides improved connectivity between the top and bottom halves. The SiP switches are re-arrangeably non-blocking MZI based 4×4 Benes topology, with 6 individual MZI elements in each switch. For this work, both switches performs as a 2×2, biased to either bar or cross states. The biasing for these configurations are shown in Fig. 4(a) and 4(b). Extinction ratios for both switches range between 10–15 dB.

5. Experimental results

The proposed control plane was evaluated by measuring the end-to-end latency experienced by packets during a synchronized circuit and packet switching operation. This latency is a combination of software control plane based latencies, and hardware latencies from the EPS and



(a) SiP Switch 1 (b) SiP Switch 2

Fig. 4. Bar and cross configurations for both SiP switch elements

SiP switching time. A visual depicting these latencies is shown in Fig. 5, and listed in Table 3 as well. First, the control plane latencies consist of 1) the flow insertion latency, 2) the time for the SDN controller to send a SiP flow update message to the FPGA Controller, 3) the time taken for FPGA Controller to generate an Ethernet packet and send it to the FPGA microcontroller, and 4) the time for the FPGA microcontroller to process the packet and apply the voltage to the DACs. The flow insertion latency from the Ryu SDN controller to a Pica8 switch was evaluated by sending 800 flows in parallel and an average of 78.5 µs per flow was observed. Next, 223 µs was measured for one-half of the Round Trip Time (RTT) of the TCP socket connection that was used for the SDN controller to send a flow update message to the FPGA Controller. This delay occurs in parallel with the EPS flow update latency. The FPGA Controller then takes 702 ns to form the Ethernet packet and send it to the FPGA microcontroller. The latency required for the FPGA microcontroller to process the packet and apply the voltages was measured to be 120 µs, which was found by viewing the Signal Tap logic analyzer and finding the difference between the moment that the flow update packet was received to the moment that the voltages were written to the DACs. The delay between sending multiple electronic control signals simultaneously in software was also measured. Fig. 6 shows that the four electronic control signals overlap and the delay between them is negligible in real-time.

The hardware latencies consist of 1) the layer 3 switching time of the Pica8 EPSs, and 2) the SiP switching time. The layer 3 switching time of the EPS was measured by performing a data transfer between Servers 2 and 3 on an indirect path, and changing it to a direct path with the same number of hops, and tracking the number of dropped packets with the program *tcpdump*, shown in Fig. 7(a) [31]. In this fig, each dot represents a single packet in time. The output of tcpdump shows the timestamp of each packet that was sent which is graphed in the x-axis, and the window size at the receiver side over this transmission in the y-axis. The window size itself is not an important metric, but the focus of this graph is to show the gap in the middle of the graph which represents the link unavailability time. The SiP switching time was measured for both bar to cross and vice versa for both switches, which are shown from Fig. 8(a) to 8(d). Each of these were measured to be 3.5 µs, except for the cross to bar operation of switch 2, which was measured to be 6 µs. It is larger than the other response times due to inconsistencies during fabrication of the device.

We measured the total control plane latency to be 344 µs, which begins from the moment that the SDN controller begins the switching operation by sending flow updates to the EPS and configuration message to the FPGA Controller, to the point where the FPGA microcontroller

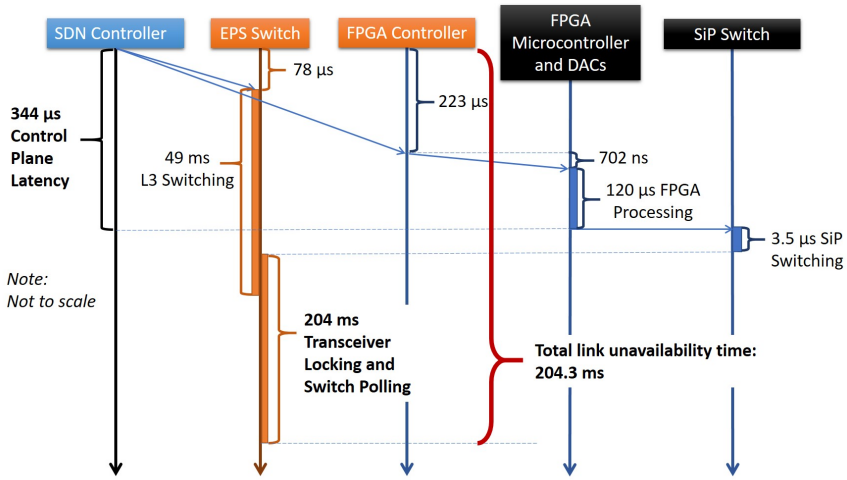


Fig. 5. Timeline showing control plane and hardware latencies

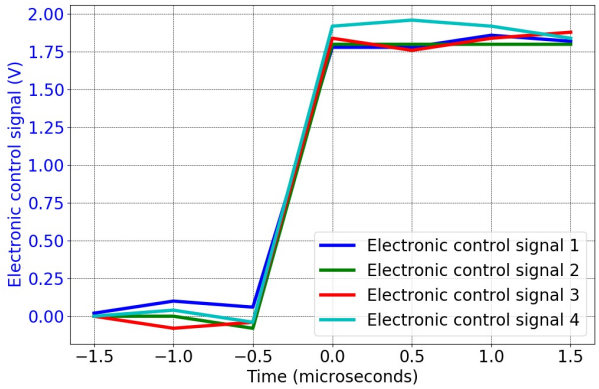
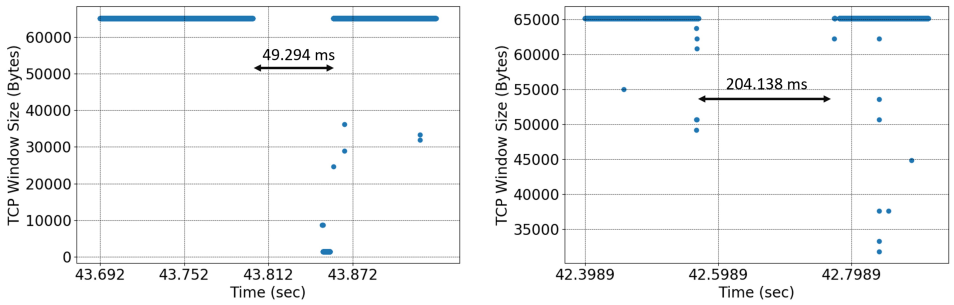


Fig. 6. Delay between four electronic signals sent simultaneously

has applied the voltage to the DACs. Having evaluated all the individual delays of the switching latency, we performed a switching operation on the testbed during a 10 Gbps data transfer and monitored the end-to-end switching latency by observing the total packets dropped, using the software tool *iperf* to perform the data transfer [32]. This was done by initially sending data from Server 1 to Server 4 with Switch 1 in the bar configuration, so that the path was an indirect route that passed through EPS 3 before reaching Server 4. Following this, Switch 1 was set to the cross configuration which provides a direct path from Server 1 to 4. Fig. 7(b) shows the *tcpdump* traces for this operation, and show that no packets were transmitted for a duration of 204.138 ms. The reason for this large delay of an extra 203.794 ms (obtained by subtracting the latency of the control plane from the total end-to-end switching latency) is due to the transceiver locking and switch polling time, which refers to the time taken by the transceivers at the source and destination ports connected to Server 1 and 4 to recognize each other's signals and to configure their mode of operation over a link, as well as the time for the Pica8 switch to poll the status of the transceivers and to report the link up status that allows the data stream to flow. This delay is discussed in the next section.

Tab 3. List of all software and hardware latencies

Control Plane Latency		**Hardware Latency**	
SDN controller per flow update	78 µs (parallel 1)	EPS L3 Switching	49 ms (parallel 2)
SDN controller sending SiP flow update message to FPGA Controller	223 µs (parallel 1)	SiP Switching	3.5 µs
FPGA Controller to process message and to send packet to FPGA microcontroller	702 ns		
FPGA microcontroller to process the packet and apply bias signals to FPGA DACs	120 µs		
Total Control Plane		344 µs	
Transceiver locking and switch polling		204 ms (parallel 2)	
Total end-to-end switching		204.3 ms	



(a) (b)

Fig. 7. (a) L3 flow insertion time on an OpenFlow EPS measured by monitoring packet transfer using *tcpdump*, (b) End-to-end switching time measured by monitoring packet transfers using *tcpdump*

Lastly, to demonstrate the potential benefit in data throughput over the network by our control plane, we show network optimization through bandwidth steering for each SiP switch individually and in a combined operation. Fig. 9(a) shows throughput over time for Servers 1 and 2. Initially, Switch 1 is in the bar configuration, and Server 1 is transmitting to Server 4 via the path EPS 3 - EPS 10 - EPS 5 - EPS 4. Meanwhile, Server 2 is transmitting to Server 5 via the path EPS 2 - EPS 3 - EPS 10 - EPS 5. This causes the two data streams to share the same path from EPS 10 to EPS 5, which causes their throughput to be limited to half the link capacity, or approximately 5 Gbps. After 6 seconds, the state of Switch 1 is changed to cross, which allows Server 1 to send data to Server 4 directly (through EPS 1 and EPS 4), while the route for Server 2 to transmit to Server 5 remains unchanged. Now that these data streams have their own dedicated inter-group link, both senders can transmit at near the full link capacity.

A similar scenario is shown for Fig. 9(b), with Switch 2 initially in the bar configuration. In this case, Server 6 is transmitting data to Server 8 via EPS 7 and EPS 8, while Server 7 is transmitting to Server 9 via EPS 8 and EPS 9. Therefore they share the link between EPS 7 and EPS 8, and transmit at a limited throughput of approximately 5 Gbps. By configuring Switch 2 into the cross state, Server 6 has a direct inter-group link to Server 8 and Server 7 has a separate dedicated link to Server 9, allowing both of the data streams to transmit near full link capacity.

Fig. 9(c) shows the same initial and final configurations as described previously, but with both switches performing a bar-to-cross switching operation simultaneously. We can observe that the throughputs in both cases change from the limited rate to near full link capacity at the same time, which shows that our control plane performs its task of controlling multiple SiP switches along with the associated flow deletions and insertions correctly. The fluctuations in the throughput

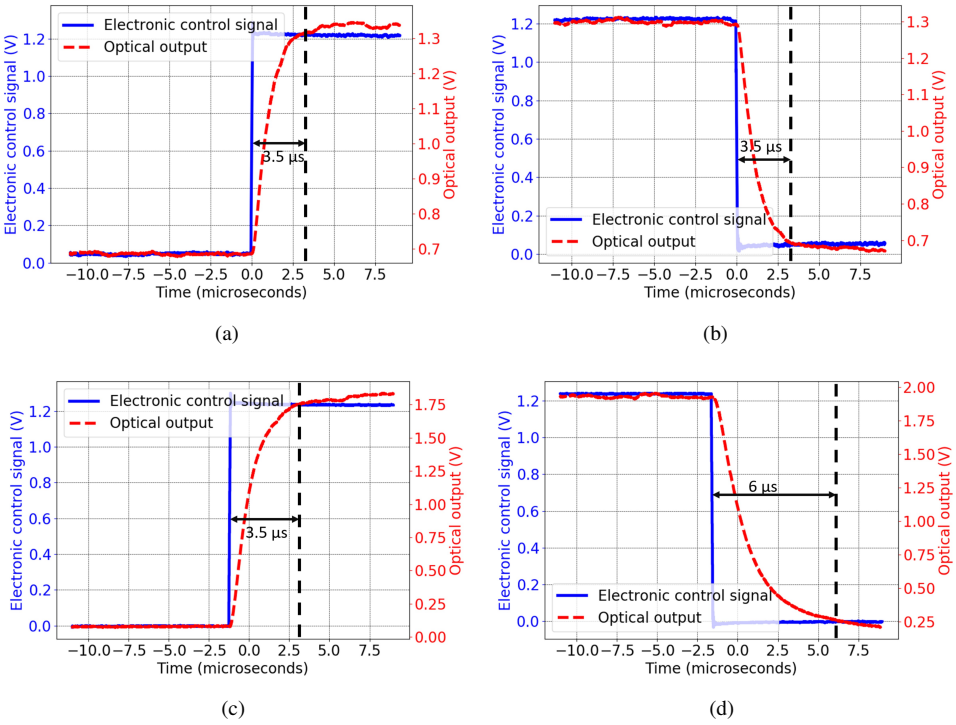
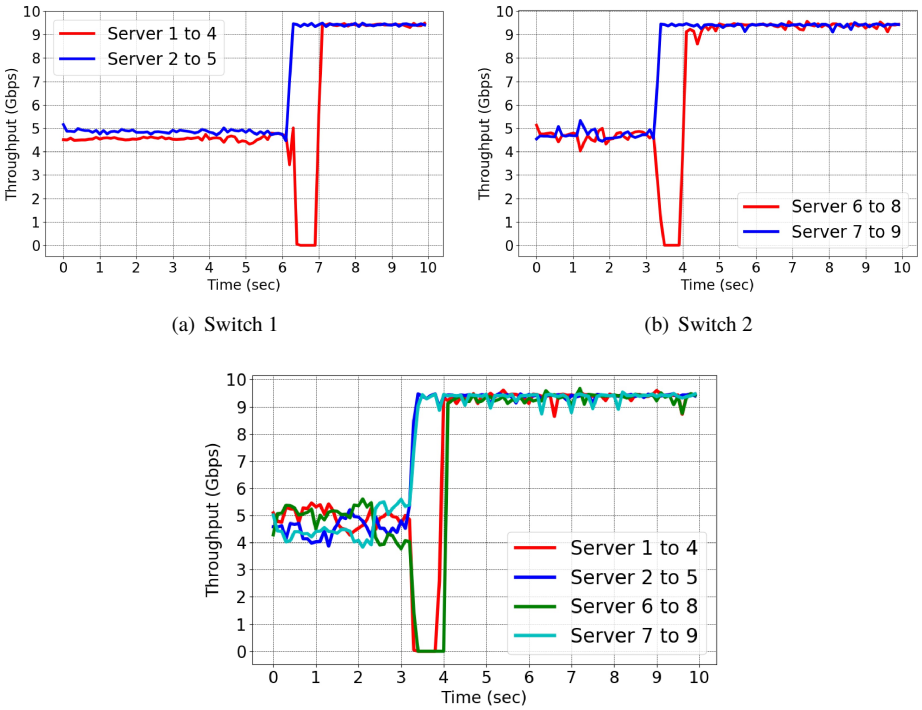


Fig. 8. MZI SiP ON and OFF switching time due to input control voltage for switch 1 (a, b) and switch 2 (c, d)

while the flows are shared in the first few seconds compared to that of the previous Figs. are purely random and do not have any significance.

6. Discussion

The SiP switch with a 3.5 µs switching time is satisfactory for physical layer reconfiguration with an approximate 50 ms latency for the EPS OpenFlow flow update and 204 ms transceiver locking and switch polling delay. The main challenge discovered in our approach is the transceiver locking and switch polling delay, which dominates overall link unavailability time. Delay times vary for different equipment models and data rates, but typical transceiver locking times range in the tens of microseconds, comparable to the switching latency of the SiP switch, while the switch polling delay for various EPSs are in the hundreds of millisecond range. Because commercial EPS vendors work with physical wiring of input and output ports to the EPS that do not change, and are not able to manipulate the polling rate in the EPS's operating system, current EPSs are ill-equipped to handle scenarios where devices such as SiP switches that manipulates the optical signal to reconfigure the physical topology thus causing frequent link state up and down changes are used alongside them. However, the development of commercial EPSs with microsecond polling time will allow for the reduction of the overall end-to-end switching latency to sub-millisecond ranges that is equivalent to a few hundred Kb packet drops on a 10G link for online switching. On the other hand, the L3 flow update latency can be mitigated by using EPSs that specialize in fast switching speeds, such as an InfiniBand switch which features microsecond scale switch latency [33]. Additionally, the development of commercial burst-mode receivers with microsecond transceiver locking time will also reduce the end-to-end latency. For



(a) Switch 1

(b) Switch 2

(c) Switch 1 and 2 combined

Fig. 9. Demonstration of throughput increase by bandwidth steering on (a) SiP switch 1, (b) SiP switch 2, and (c) simultaneous switching on both SiP switches

example, [35] has demonstrated nanosecond locking time burst-mode transceivers with data rates up to 25 Gbps. As we move from static topologies to optical networks with more flexibility and dynamic switching, EPS manufacturers will deploy ToR systems with lower link up times that are comparable to the switching times of the SiP devices. In the meantime, fast electronic ToR switching that can operate alongside SiP devices will have to be made with custom FPGA designs. However, this falls outside the scope of this work, which focuses on the integrability of SiP switching devices with current Datacom network architectures.

Despite the millisecond range link unavailability time due to the current EPS polling rate, optical circuit switching can still greatly benefit current DC and HPC applications that have well-known traffic characteristics between neighbors of nodes. For example, in HPC topologies where the SiP switch is used to interconnect between different groups of nodes as demonstrated in the experiments, it is unlikely that the HPC application require rapid circuit switching as the traffic characteristics of HPC applications do not vary much over time; rather, optical circuit switching will be used in a mostly preliminary manner to configure the network topology to optimally benefit the application's traffic matrix before the application's actual runtime, as we have demonstrated in [1]. The average runtime of multiple HPC applications occupying an HPC system are in the range of hours, which means that an end-to-end switching delay in the hundreds of millisecond range is acceptable.

Overall, our proposed network architecture with integration of SiP switching elements is designed for both DC interconnection networks and HPC networks is highly applicable to high-bandwidth optical circuit switching, because of the switches inherent data transparency property. With an SDN controlled SiP switching fabric that works alongside conventional

electronic networks, we can provide the agility and automation that delivers efficient network resource usage through bandwidth steering at lower cost and improved compatibility. Because we use a distributed FPGA network as the control hardware for controlling SiP devices, these devices can be swapped out for any other active network element as long as they are controlled by applying bias signals, including both MZI and micro-ring resonator based switches of varying radices [34]. As long as one knows the bias voltage values that are required to configure any SiP device, then such a device can be used in our network architecture. The SDN controller does not need to know about the device physics, but only needs to know which possible configurations that the connected device can be set to. For larger SiP switch fabrics, information regarding routing within a SiP switch matrix can be approached from two sides, i.e. the capability of the connectivity and routing strategy. First, the capability of the connectivity of switch fabrics depends on the blocking characteristics of the applied architectures. This can be generally classified as blocking, rearrangeably non-blocking, wide-sense non-blocking, strictly non-blocking and redundant strictly non-blocking. The increase of the capability of the connectivity is at the cost of using more switching elements. To achieve a full connection for an input/output permutation, non-blocking architectures are required. In this paper, we applied the rearrangeably non-blocking Benes topology. Second, for a given switch, the switch elements and shuffle networks can be represented by transition matrixes and therefore the routes can be fully pre-calculated. We have developed a simulation platform to do a full calculation on the Benes switch with mathematical analysis on the path routing variations in [36]. This further helps generating a smart routing table for switch control [30].

7. Conclusion

The employment of SiP switches for optical circuit switching can vastly improve the performance and efficiency of DC and HPC networks. However, the integration of SiP with electronic network architectures requires a control plane capable of synchronized switching with minimal packet drop. We present a scalable SDN control plane that integrates multiple SiP switches with conventional Ethernet networks used in current DC and HPC architectures. It features unified management of both electronic and SiP network elements by performing OpenFlow enabled flow modification on the EPSs and uses an in-house FPGA Controller application together with a distributed FPGA network to manage SiP active elements, allowing it to perform simultaneous packet and circuit switching. Our experimental Dragonfly testbed allowed us to evaluate each individual latency associated with the various software and hardware interfaces involved in a switching operation as well as demonstrate network optimization through bandwidth steering on two SiP switches in parallel. We measured an overall 344 µs latency for our proposed control plane.

For our future work we will further explore the applicability of SiP devices in HPC systems by integrating SiP devices on an InfiniBand network, demonstrate improvements in performance of HPC benchmark applications, and seek possible solutions to improve the synchronicity between the control plane and control hardware.

Funding

CIAN NSF ERC (EEC-0812072); NSF NeTS (CNS-1423105); Air Force Research Laboratory (FA8650-15-2-5220); U.S. Department of Energy (DoE) Advanced Simulation and Computing (ASC) program through contract with Sandia National Laboratories (PO 1319001); Advanced Research Projects Agency - Energy (ARPA-E) (DE-AR0000843).