

High Performance Silicon Photonic Interconnected Systems

Ziyi Zhu

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2022

© 2022

Ziyi Zhu

All Rights Reserved

Abstract

High Performance Silicon Photonic Interconnected Systems

Ziyi Zhu

Advances in data-driven applications, particularly artificial intelligence and deep learning, are driving the explosive growth of computation and communication in today's data centers and high-performance computing (HPC) systems. Increasingly, system performance is not constrained by the compute speed at individual nodes, but by the data movement between them. This calls for innovative architectures, smart connectivity, and extreme bandwidth densities in interconnect designs. Silicon photonics technology leverages mature complementary metal-oxide-semiconductor (CMOS) manufacturing infrastructure and is promising for low cost, high-bandwidth, and reconfigurable interconnects. Flexible and high-performance photonic switched architectures are capable of improving the system performance. The work in this dissertation explores various photonic interconnected systems and the associated optical switching functionalities, hardware platforms, and novel architectures. It demonstrates the capabilities of silicon photonics to enable efficient deep learning training.

We first present field programmable gate array (FPGA) based open-loop and closed-loop control for optical spectral-and-spatial switching of silicon photonic cascaded micro-ring resonator (MRR) switches. Our control achieves wavelength locking at the user-defined resonance of the MRR for optical unicast, multicast, and multiwavelength-select functionalities. Digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are necessary for the control of the switch. We experimentally demonstrate the optical switching functionalities

using an FPGA-based switch controller through both traditional multi-bit DAC/ADC and novel single-wired DAC/ADC circuits. For system-level integration, interfaces to the switch controller in a network control plane are developed. The successful control and the switching functionalities achieved are essential for system-level architectural innovations as presented in the following sections.

Next, this thesis presents two novel photonic switched architectures using the MRR-based switches. First, a photonic switched memory system architecture was designed to address memory challenges in deep learning. The reconfigurable photonic interconnects provide scalable solutions and enable efficient use of disaggregated memory resources for deep learning training. An experimental testbed was built with a processing system and two remote memory nodes using silicon photonic switch fabrics and system performance improvements were demonstrated. The collective results and existing high-bandwidth optical I/Os show the potential of integrating the photonic switched memory to state-of-the-art processing systems. Second, the scaling trends of deep learning models and distributed training workloads are challenging network capacities in today's data centers and HPCs. A system architecture that leverages SiP switch-enabled server regrouping is proposed to tackle the challenges and accelerate distributed deep learning training. An experimental testbed with a SiP switch-enabled reconfigurable fat tree topology was built to evaluate the network performance of distributed ring all-reduce and parameter server workloads. We also present system-scale simulations. Server regrouping and bandwidth steering were performed on a large-scale tapered fat tree with 1024 compute nodes to show the benefits of using photonic switched architectures in systems at scale.

Finally, this dissertation explores high-bandwidth photonic interconnect designs for disaggregated systems. We first introduce and discuss two disaggregated architectures leveraging extreme high bandwidth interconnects with optically interconnected computing resources. We present the concept of rack-scale graphics processing unit (GPU) disaggregation with optical circuit switches and electrical aggregator switches. The architecture can leverage the flexibility of high bandwidth optical switches to increase hardware utilization and reduce application runtimes.

A testbed was built to demonstrate resource disaggregation and defragmentation. In addition, we also present an extreme high-bandwidth optical interconnect accelerated low-latency communication architecture for deep learning training. The disaggregated architecture utilizes comb laser sources and MRR-based cross-bar switching fabrics to enable an all-to-all high bandwidth communication with a constant latency cost for distributed deep learning training. We discuss emerging technologies in the silicon photonics platform, including light source, transceivers, and switch architectures, to accommodate extreme high bandwidth requirements in HPC and data center environments. A prototype hardware innovation - Optical Network Interface Cards (comprised of FPGA, photonic integrated circuits (PIC), electronic integrated circuits (EIC), interposer, and high-speed printed circuit board (PCB)) is presented to show the path toward fast lanes for expedited execution at 10 terabits.

Taken together, the work in this dissertation demonstrates the capabilities of high-bandwidth silicon photonic interconnects and innovative architectural designs to accelerate deep learning training in optically connected data center and HPC systems.

Table of Contents

Acknowledgments	xii
Dedication	xiii
Chapter 1: Introduction and Background	1
1.1 Deep Learning Trends and Challenges	1
1.2 Silicon Photonic Circuit Switching	7
1.3 Silicon Photonic High Bandwidth Transceivers	10
1.4 Scope of Dissertation	15
Chapter 2: FPGA-controlled Silicon Photonic Interconnects	17
2.1 Introduction	17
2.2 Optical Switching in Cascaded-MRR Switch	18
2.3 FPGA-Based Open Loop Control	20
2.3.1 System Architecture	20
2.3.2 Experimental Demonstration for Unicast and Multicast	22
2.3.3 Experimental Demonstration for Multiwavelength-select	24
2.4 FPGA-Based Closed Loop Control	26
2.4.1 Closed-loop Single-wire DAC and ADC Architecture	27
2.4.2 Experimental Setup	29

2.4.3	Results	30
2.5	Chapter Summary	33
Chapter 3: Optically Connected Memory for Deep Learning		34
3.1	Introduction	34
3.2	System Architecture	36
3.2.1	(de)Serialization of Memory Transfers	38
3.2.2	Map to Local System Address Space	39
3.2.3	SiP Switch Control	39
3.2.4	Accelerator Design	41
3.3	Testbed	42
3.4	Experiment and Results	45
3.4.1	Optical Spectra	45
3.4.2	Eye Diagrams	47
3.4.3	Switching Time	48
3.4.4	End-to-end Reconfiguration Time	49
3.4.5	Application and Execution Time	50
3.5	Discussion	54
3.6	Chapter Summary	57
Chapter 4: Photonic Switched Architectures for Distributed Deep Learning		58
4.1	Introduction	58
4.2	System Architecture and SiP Switch Control	60
4.2.1	System Architecture	60

4.2.2	SiP Switches and Control	61
4.3	Testbed	63
4.4	Experiments and Results	66
4.5	System-scale Evaluation	69
4.5.1	Simulation Setup	70
4.5.2	Server Regrouping and Bandwidth Steering	71
4.5.3	Results	72
4.6	Chapter Summary	76
Chapter 5:	Disaggregated Architectures for Deep Learning	77
5.1	Flexible Optically Interconnected Computing Resources	77
5.1.1	Introduction	77
5.1.2	System Architecture	78
5.1.3	Experimental Setup and Results	80
5.1.4	Conclusion	83
5.2	SiPAC: Silicon Photonic Accelerated Compute Cluster	83
5.2.1	Introduction	83
5.2.2	Topology Design	85
5.2.3	Wavelength Routing/Selection	86
5.2.4	Testbed Experiment	88
5.2.5	System-scale Simulation	91
5.2.6	Conclusion	93
5.3	Chapter Summary	94

5.3.1	Future Work	94
Chapter 6:	Fast Lanes for Expedited Execution at 10 Terabits	96
6.1	Introduction	96
6.2	ONIC Module Overview	97
6.2.1	Overall Packaging Plan	97
6.2.2	Link Analysis	99
6.3	Silicon Photonic Transceiver Chip	100
6.4	Development of O-NIC PCB	103
6.5	Test Packages	104
6.5.1	RX Test Package	105
6.5.2	TX Test Package	110
6.6	PCIe Interface	110
6.7	Chapter Summary	113
Conclusion	114
References	117

List of Figures

1.1	Model Trend from 2016 to 2022.	2
1.2	Data parallelism, tensor model parallelism, pipeline model parallelism, and hybrid parallelism.	3
1.3	Illustrations of localized nodes and fragmented nodes	5
1.4	Hardware bandwidth trend from 2015 to 2022.	6
1.5	(a) Low insertion loss 32×32 MZI switch, reprinted from [47]. (b) Polarization-diversity 32×32 MZI switch, reprinted from [48]. (c) 64×64 T-O MZI switch, reprinted from [49]. (d) 16×16 E-O MZI switch, reprinted from [50]. (e) 32×32 E-O MZI switch, reprinted from [51]. (f) Hybrid 2×2 MZI-SOA module, reprinted from [52].	8
1.6	(a) 8×7 cross-bar switch, reprinted from [53]. (b) 8×8 Omega switch, reprinted from [54]. (c) 4×4 switch-and-select switch, reprinted from [55]. (d) 4×4 hitless switch, reprinted from [56].	9
1.7	(a) 240×240 MEMS cross-bar switch, reprinted from [60]. (b) Polarization-Insensitive MEMS switch, reprinted from [61]. (c) 32×32 MEMS switch fabricated in a commercial foundry, reprinted from [62].	11
1.8	The various integration approaches to integrate PICs with EICs. (a) Monolithic integration. (b) 2D integration. (c) 3D integration. (d) 3D integration with active photonic interposer. (e) 2.5D integration. Reprinted from [63].	11
1.9	TeraPHY chiplet showing optical Tx/Rx macros, fiber array, and Tx/Rx circuits. Reprinted from [68].	13
1.10	(a) Schematic of the comb-driven transceiver architecture, reprinted from [69]. (b) Micrograph of the full die, reprinted from [69]. (c) Image of the Kerr comb device, reprinted from [70].	14

2.1	(a) 1×2 MZI cell. (b) 1×2 MRR cell. (c) 1×2 MEMS-actuated coupler cell. (d) Packaging with wirebonding and fiber attachment	18
2.2	(a) Switch architecture with 8 Cascaded MRRs. (b) Zero bias scenario. (c) Unicast scenario. (d) Multicast scenario. (e) Multiwavelength-select scenario.	19
2.3	Open-loop control scheme.	21
2.4	(a) Tunable laser. (b) Schematic of the unicast and multicast experimental setup. (c) FPGA-based switch controller and the packaged MRR switch on a PCB.	22
2.5	Experimental results of 8 different cases including unicast, multicast, and broadcast.	23
2.6	Schematic of the multiwavelength-select experimental setup.	24
2.7	Experimental results for multiwavelength-select operations.	25
2.8	(a) Schematic of digital-to-analog and analog-to-digital control and feedback (b) Transient of digital-to-analog circuit response and MRR response due to a changed duty cycle of the PWM control signal.	27
2.9	Experimental setup demonstrating the single-wire ADC and DAC silicon photonic circuits evaluated with PAM-4 signal.	29
2.10	(a) Transient responses for unicasting of MRR #1. (b) Corresponding spectrums and eye diagrams of MRR #1 before and after tuning. (c) Transient responses for unicasting of MRR #2. (d) Corresponding spectrums and eye diagrams of MRR #2 before and after tuning.	30
2.11	(a) Optical transient before and after the feedback tuning procedure for multicasting. (b) Received PWM transient. (c) Corresponding spectrum and eye diagrams before the tuning procedure. (d) Corresponding spectrum and eye diagrams after tuning.	32

3.1	(A) On the left, the traditional system architecture with each processing system composed of preconfigured and fixed CPU, memory, storage, accelerator and network resources. In our proposed system architecture, on the right, each processing system using optical I/Os is also connected to a remote memory pool through photonic interconnects. (B) Detailed implementation of photonic switched system architecture with optically connected memory. The processing system includes additional (de)serialization and transceiver (XCVR) helper blocks for (de)serializing memory mapped transactions being transmitted through optical links. On the right, remote double data rate synchronous dynamic random-access memory (DDR) nodes, are also equipped with the (de)serialization and XCVR helper blocks, and the photonic interconnects physically connect remote memory nodes to the processing system.	37
3.2	(A) An example of case 1, two remote memory resources mapped to two AXI chip2chip cores in the local processing system for the unswitched case after the resources are assigned. Each chip2chip core is assigned with a unique memory address offset (B) An example of case 2, the switching case. Both remote DDR #1 and remote DDR #2 are mapped to the AXI chip2chip #1 in the processing system. They share the same memory address offset.	40
3.3	SiP switches' configurations for the dynamic access to remote DDRs. (A) Remote memory resources to the processing system direction. (B) The processing system to remote memory resources direction.	42
3.4	(A) Experimental setup demonstrating a case of photonic switched optically connected memory system with dynamic allocation of remote DDR resources to the processing system. (B) Key hardware components. One Xilinx ZCU106 board containing the processing system and the remote DDR #1 nodes, another ZCU106 board containing only the remote DDR #2 node, and the TR4 switch controller FPGA board. (C) A packaged SiP MRR based switch with electrical SMA interface.	44
3.5	Optical spectra at the drop port of each MRR for two different configurations. (A) Two SiP switches configured as the processing system connecting to the remote DDR #1 node. (B) Two SiP switches configured as the processing system connecting to the remote DDR #2 node. (In this figure, the MRR numbers are consistent with the MRR numbers shown in Fig. 3.3.)	46
3.6	Screen shots of open eye diagrams of connected receiver ports at 10 Gb/s PRBS-31. (A) Two SiP switches configured as the processing system (PS) connecting to the remote DDR #1 node. (B) Two SiP switches configured as the processing system connecting to the remote DDR #2 node.	47

3.7	Screen shots of open eye diagrams of connected receiver ports at 10 Gb/s PRBS-31. (A) Two SiP switches configured as the processing system (PS) connecting to the remote DDR #1 node. (B) Two SiP switches configured as the processing system connecting to the remote DDR #2 node.	48
3.8	Loading/storing latencies using hard drive and remote DDR memory of different batch sizes.	51
3.9	Timelines comparing system latencies in different scenarios for switching case #2. .	54
4.1	(a) System architecture demonstration with server nodes arranged in the fat tree topology to show SiP switch-based server regrouping and higher-layer bandwidth steering. (b) An example of before (left) and after (right) server regrouping. (c) An example of before (left) and after (right) bandwidth steering above the ToR.	60
4.2	(a) Overall network control plane. (b) SiP OCS subsystem including the SiP network controller, SiP switch controller, and SiP switches. (c) The SiP network controller board (ZCU106), SiP switch controller board (TR4), and PCB holding a packaged SiP switch.	61
4.3	A 16-node experimental testbed with SiP OCSs and EPSs in a reconfigurable fat tree topology.	62
4.4	Experimental setup demonstrating the cases of server regrouping and bandwidth steering above the ToR.	63
4.5	A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller.	64
4.6	A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller.	65
4.7	Throughput of the links to server #9 and #10, from EPS #1 to EPS #5, and from EPS #5 to EPS #7 for test case #1 in the synchronized training of the VGG neural network.	67
4.8	Throughput of the links for the test case #2 in the synchronized training.	67
4.9	Throughput of the links for the test case #1 in the asynchronized training.	68
4.10	Throughput of the links for the test case #2 in the asynchronized training.	68
4.11	A 1024-node untapered fat tree topology with SiP OCSs in between the server-ToR and ToR-aggregation layers.	70

4.12	Average flow throughput of all the flows as a function of the tapering ratio for all the traffic and job mapping scenarios. RG denotes regrouping and BS denotes higher-layer bandwidth steering. With the exception of Uniform (uniform job-mapping), all other cases assume an adversarial interpod job mapping as described in the Simulation Setup Section. (a) Results for ring all-reduce flows. (b) Results for parameter server flows.	74
5.1	(a) Traditional architecture for GPU clusters. (b) Proposed system architecture for disaggregated GPU resources.	78
5.2	(a) Network control plane. (b) Testbed architecture. (c) Hardware implementations using Calient MEMS switch, server rack with ToR switch and rack server.	79
5.3	Testbed topology for two synchronous ML workload using 4 GPUs and 2 GPUs for training. Experimental results that show network performance and utilization improvements.	81
5.4	Testbed topology for two synchronous ML workload using 3 GPUs and 3 GPUs for training. Experimental results that show network performance and utilization improvements.	82
5.5	a) SiPAC architecture based on the recursive BCube topology with $L = l + 1$ levels. The l th level is constructed from r^l r -port switches and $r(l - 1)$ th level units. b) The base unit of the SiPAC topology where r CUs are connected to a WSS.	85
5.6	Example of wavelength multiplexing for a 3×3 multi-WSS and 8 wavelengths per transmitter ($w = 8$).	87
5.7	(a) Experimental setup for the comb laser. (b) Micrograph of Kerr comb source. (c) Output of Kerr Comb source. (d) Micrograph of packaged 1×8 microring resonator switch.	88
5.8	(a) Output spectrum of Ring 1. (b) Output spectrum of Ring 2. (c) Inset of (a), illustrating the sidebands surrounding the modulated carrier. (d) Output eye-diagrams of dropped signals.	90
5.9	Job completion time of different topology-collective combinations for varying network sizes. The message size is set to be 1MB.	92
6.1	FLEET hardware overview, reprinted from [141].	97
6.2	FLEET O-NIC packaging plan.	98

6.3	FLEET O-NIC link and power budget.	99
6.4	FLEET PIC transmitter and receiver banks.	100
6.5	FLEET PIC floor plan. (a) The two 16-channel transceiver links. (b) The transmitter and receiver banks. (c) Two 8-channel transceiver banks and test structures. . . .	101
6.6	(a) FLEET PIC chip.(b) Illustration of a populated interposer.	102
6.7	(a) FLEET PIC chip.(b) Illustration of a populated interposer.	103
6.8	FLEET O-NIC PCB layout.	104
6.9	FLEET RX test PIC chip and packaging plan.	105
6.10	TIA bond pad assignment.	106
6.11	(a) Interposer differential traces. (b) ADS layout model. (c) S_{21} and S_{11} results. . .	107
6.12	(a) ADS layout model for PCB differential to single ended traces. (b) ADS circuit model. (c) S_{21} and S_{11} results.	107
6.13	(a) ADS EMPro edge sma model. (b) 3D model with landing trace. (c) S_{21} result. (d) S_{11} result.	108
6.14	(a) ADS 3D wirebonds model. (b) S_{21} result. (c) S_{11} result.	108
6.15	(a) RX test package full assembly. (b) Wire-bonding picture. (c) Eye diagrams. . .	109
6.16	(a) TX test PIC chip. (b) TX test board. (c) Packaging plan with a driving EIC. . .	110
6.17	(a) Schematic of the system for the host and FPGA memory transfers. (b) Physical hardware.	111
6.18	PCIe subsystem block diagram.	112

List of Tables

3.1	System Performance Measurements	53
4.1	Experiment and Simulation Performance Measurements	75

Acknowledgements

First and foremost, I would like to thank my advisor, Professor Keren Bergman, for giving me the opportunity to join the Lightwave Research Laboratory at Columbia University. Her guidance and support throughout my graduate studies helped me not only build my knowledge and skills, but also shape my viewpoint of conducting research.

I want to thank my colleagues at the Lightwave Research Laboratory, especially Dr. Madeleine Glick, Dr. Qixiang Cheng, and Dr. Alexander Gazman for their mentorship and advise during my doctoral study. I am also grateful to other members in the group including Dr. Yiwen Shen, Dr. Min Yee Teh, Maarten Hattink, Richard Dai, Brian Wu, Kristoff Yan, and many others for their discussions, collaborations, and contributions to my research projects. Additionally, I would like to thank Constance Zhou for keeping me optimistic and happy during my Ph.D.

Also, I want to thank my mentors who provided the opportunities for my internships. In particular, Dr. Noriaki Kaneda at Nokia Bell Labs and Dr. Urs Muller at Nvidia. During my internships, I learned valuable skill sets, which assist my research explorations.

Further, I would like to acknowledge the support of my research projects from the Advanced Research Projects Agency Energy (ARPA-E) under the ENLITENED Project, the U.S. Department of Energy (DOE) SBIR/STTR Program under the P-SSIO Project, and the U.S. Defense Advanced Research Projects Agency (DARPA) under the FastNICs Project.

Finally, I would also like to thank my thesis committee members, Professor Keren Bergman, Professor Luca Carloni, Professor Zoran Kostić, Professor Christine Hendon, and Professor Mingoo Seok for their valuable time and their evaluation of my thesis.

Dedication

This dissertation is dedicated to my parents, Nanyuan Ke and Zhenwu Zhu.

Chapter 1: Introduction and Background

1.1 Deep Learning Trends and Challenges

Deep learning (DL) is a branch of machine learning that has drastically improved the state-of-the-art in many applications that enhance our daily lives and impact various aspects of our society. The computational models used in deep learning, called deep neural networks (DNNs), have been successfully applied to various fields including activity recognition [1], image classification [2], and natural language processing [3]. The DNNs consist of many processing layers whose computation is mainly defined by weights and biases. These weights and biases, called parameters of the DNNs, are learned during the training, and used for the inference. Accelerators, such as graphics processing units (GPUs) and field programmable gate arrays (FPGAs), are used for accelerating these training and inference processes [4, 5]. As researchers make continuous studies to improve these DL models, it is found that larger model sizes and larger dataset usually result in better model accuracy [6, 7]. Increased model size is at the center of these advancements [8, 9, 10], and multiple studies have shown that this trend will continue in natural language processing particularly [11, 12]. As a result, there has been significant investment in training huge models. As Fig.1.1 shows, the model sizes have increased over 100,000 times over the past five years to more than 1 trillion parameters.

Researchers have adopted distributed deep learning training [14, 18, 19], where the DL models or the training datasets are partitioned and distributed onto GPUs, to train such large models. There are three major parallelism strategies: data parallelism (DP), model parallelism (MP), and hybrid parallelism (HP). For a model that fits in the device memory for training, DP is used to scale training to multiple devices. DP keeps a full copy of the entire training model on each worker node while distributing the partitioned input dataset to different working nodes. Each worker executes

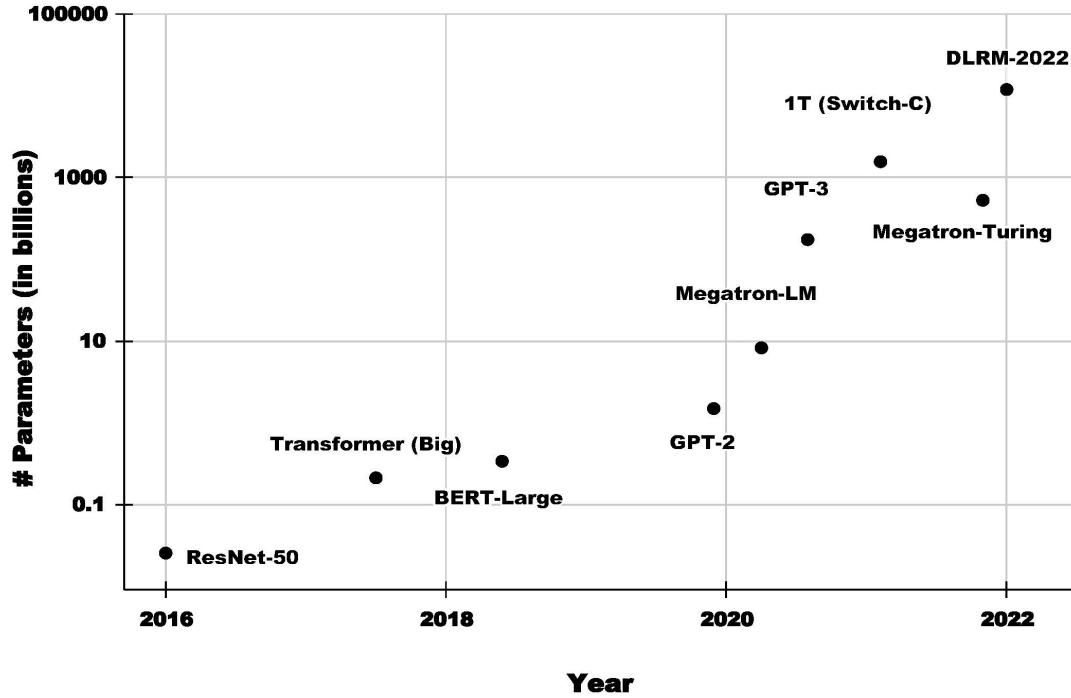


Figure 1.1: Model Trend from 2016 to 2022.
[2, 3, 8, 13, 9, 14, 15, 16, 17]

the forward and backward propagation on a different subset of data samples. For synchronized training, workers need to synchronize their averaged gradients with each other and update the model locally at each integration through all-reduce collective operation. For asynchronized training, each worker sends the averaged gradients to parameter servers [20] and retrieves the updated model parameters for each iteration without waiting for others. When a model does not fit in the device memory, tensor model parallelism (TP) [14, 21] and pipeline model parallelism (PP) [18, 22] can be performed. In TP, the training model is split vertically across layers, so each tensor layer is placed on multiple nodes. The results on each node are then synchronized across the DP workers. In PP, the training model is split horizontally across layers, placing one or more layers on each working node. This requires computing nodes to receive activations/gradients from previous layers before they can proceed onto computing the next layer. Hybrid Parallelism (HP) combines both DP and MP to partition both the models and the dataset. Computing nodes are divided into DP groups of MP nodes. At each iteration, each group of MP nodes synchronize among themselves

using MP-transfers and then synchronize across the DP groups. Figure 1.2 illustrates DP, TP, PP and HP using DP and TP, respectively.

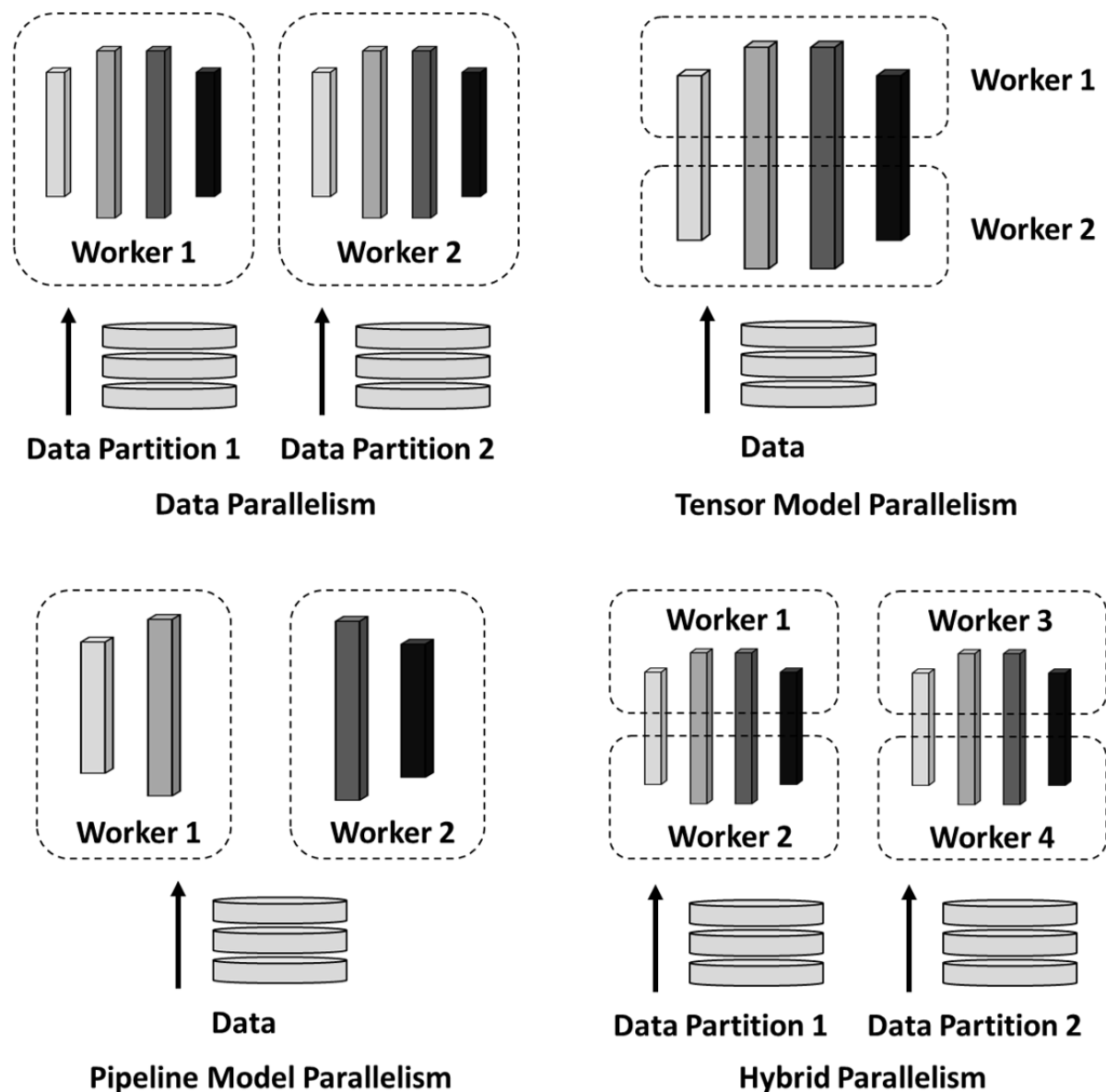


Figure 1.2: Data parallelism, tensor model parallelism, pipeline model parallelism, and hybrid parallelism.

Despite the distributed deep learning training and parallelism techniques, there are three major

challenges. While the maximum memory requirement keeps growing, the real-time memory usage is application dependent and often requires on-demand solutions. First, different deep learning applications show varying memory requirements based on their architectures for example convolutional neural networks (CNNs), recurrent neural networks (RNNs), transformers, and etc.) Second, the memory capacity requirement for various batch sizes [23] and optimization strategies [24] can change within a large range, but the method requiring a larger memory size does not always guarantee a better system performance [25]. Lastly, the size of embeddings that are used in recommendation applications is dependent on the entry size and number of models [26]. Having a fixed and preconfigured amount of memory in the local system for the maximum memory capacity requirement is inefficient and will become more so. A scalable and dynamic solution is required to address the memory challenges for future deep learning applications.

Besides, DL workloads are taking a large proportion of the computation in today's HPC operations, and observation has shown that the demand is dramatically growing in datacenters [27]. Distributed deep learning workloads can require many server nodes and show strong communication patterns between these nodes. These trends have shifted the performance bottleneck from the compute to the network interconnect due to system fragmentation (applications often receive an allocation on a set of distant and non-contiguous nodes) [28]. An example of system fragmentation in a tapered 16-node fat tree topology is shown in Fig. 1.3. Under the top-of-rack (ToR) switch the full bandwidth can be utilized under the best scenario, no traffic goes beyond the ToR switches. However, for the fragmented case, nodes for running the same job can distribute across the network and constrained bandwidth is experienced. Due to the constrained links at higher layers, training workloads can be slowed down substantially. This places a tremendous challenge on interconnect designs to provide high bandwidth and low latency networking to sustain the continual growth of these hardware-driven deep learning applications. These challenges present a unique opportunity for flexible photonic switched networks that have the capabilities to perform topology reconfiguration and have motivated much research to explore reconfigurable network architectures based on optical circuit switches (OCSs).

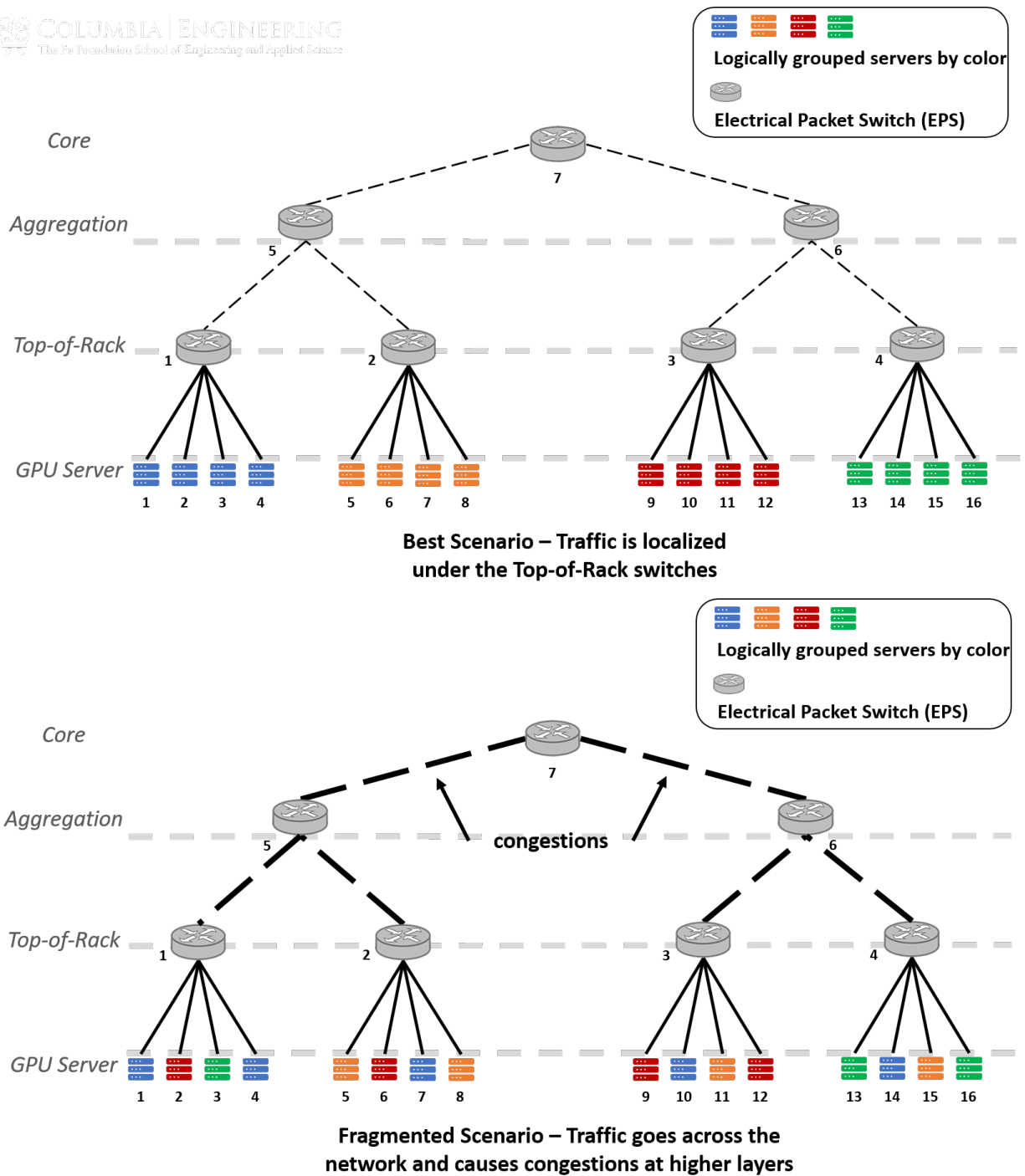


Figure 1.3: Illustrations of localized nodes and fragmented nodes

Lastly, researchers have also demonstrated the GPU clusters that have increased to more than

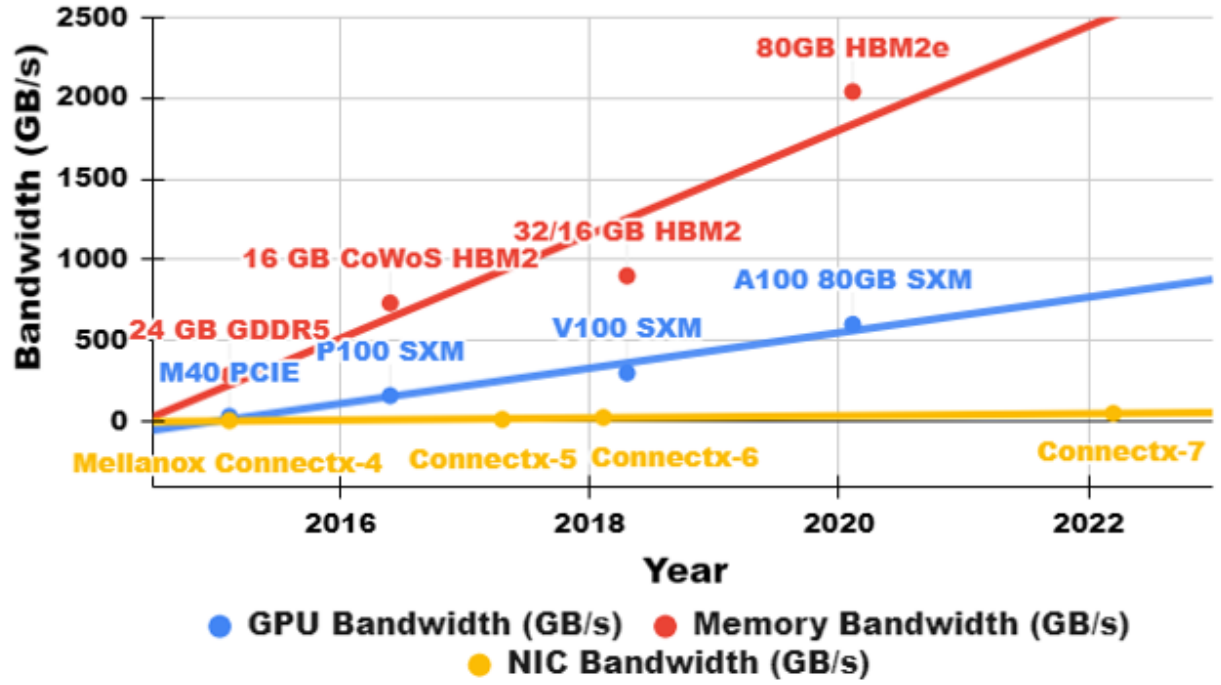


Figure 1.4: Hardware bandwidth trend from 2015 to 2022.
[30, 31, 32, 33, 34, 35, 36, 37]

3000 GPUs [29] for training extremely large models with hybrid parallelism. However, current hardware solutions can only provide high-bandwidth network for a limited group of computing units (e.g. Nvidia DGX Station connects 8-16 GPUs using high speed NVSwitches and NVLinks for up to 600 GB/s bidirectional bandwidth [30]). To scale the training to sizes larger than the group size would require inter-group communication that relies on 200 Gb/s InfiniBand links which are much slower than the intra-group fabric. And this bandwidth discrepancy has severely limited the communication efficiency during the training process. As Fig.1.4 shows, the increase in network bandwidth is not able to catch up with the exponential rate at which the model sizes are increasing. And the on-board memory/GPU bandwidth (red and blue lines in Fig.1.4 have been growing at a much faster rate than the inter-board bandwidth (yellow line).

To this end, novel reconfigurable architectures and high-bandwidth inter-node interconnects are required to address these challenges and to meet the requirement of scaling trends of deep learning training.

1.2 Silicon Photonic Circuit Switching

Optical circuit switching offers a promising approach to reconfigure the interconnect in order to address the challenges mentioned above. In particular, optical switches can (1) regroup a set of distant and non-contiguous nodes and (2) steer bandwidth at network layers for efficiency. Depending upon underlying traffic patterns of the nodes at different times, optimized topology connections can be dynamically formed on demand.

Commercially available technologies, such as microelectromechanical systems (MEMS) [38], beam-steering [39], and liquid crystal on silicon (LCOS) [40], can be used to implement the reconfigurable network. However, there are still challenges to achieve commercial adoption. The rigorous calibration and the installation of discrete components introduce significant complexity and result in high cost per port. Similarly, arrayed waveguide grating routers (AWGRs) [41] based interconnects usually require higher cost tunable wavelength transceivers that add complexity and additional power consumption in broadcast and select type architectures. For low-cost datacenter/HPC adoption, lithography-based photonic integration technologies hold great promise for large-scale optical integrated switch fabrics with smaller device footprint, and reduced assembly and calibration overheads.

The silicon photonics (SiP) platform, in particular, leverages the mature and widespread CMOS manufacturing infrastructure, and SiP switches are promising for the dynamic topology reconfiguration with better power efficiency, lower cost-per-port, smaller footprint, and the potential for nanosecond range dynamic switching [42, 43, 44, 45, 46]. However, there are several technical challenges to address in this platform, specifically loss through the switch, polarization dependency, thermal stability, and switch radix scalability. Research works have been reported to address these challenges, and the primary switching cells that are being explored are Mach-Zehnder interferometers (MZIs), microring resonators (MRRs), and MEMS-actuated couplers.

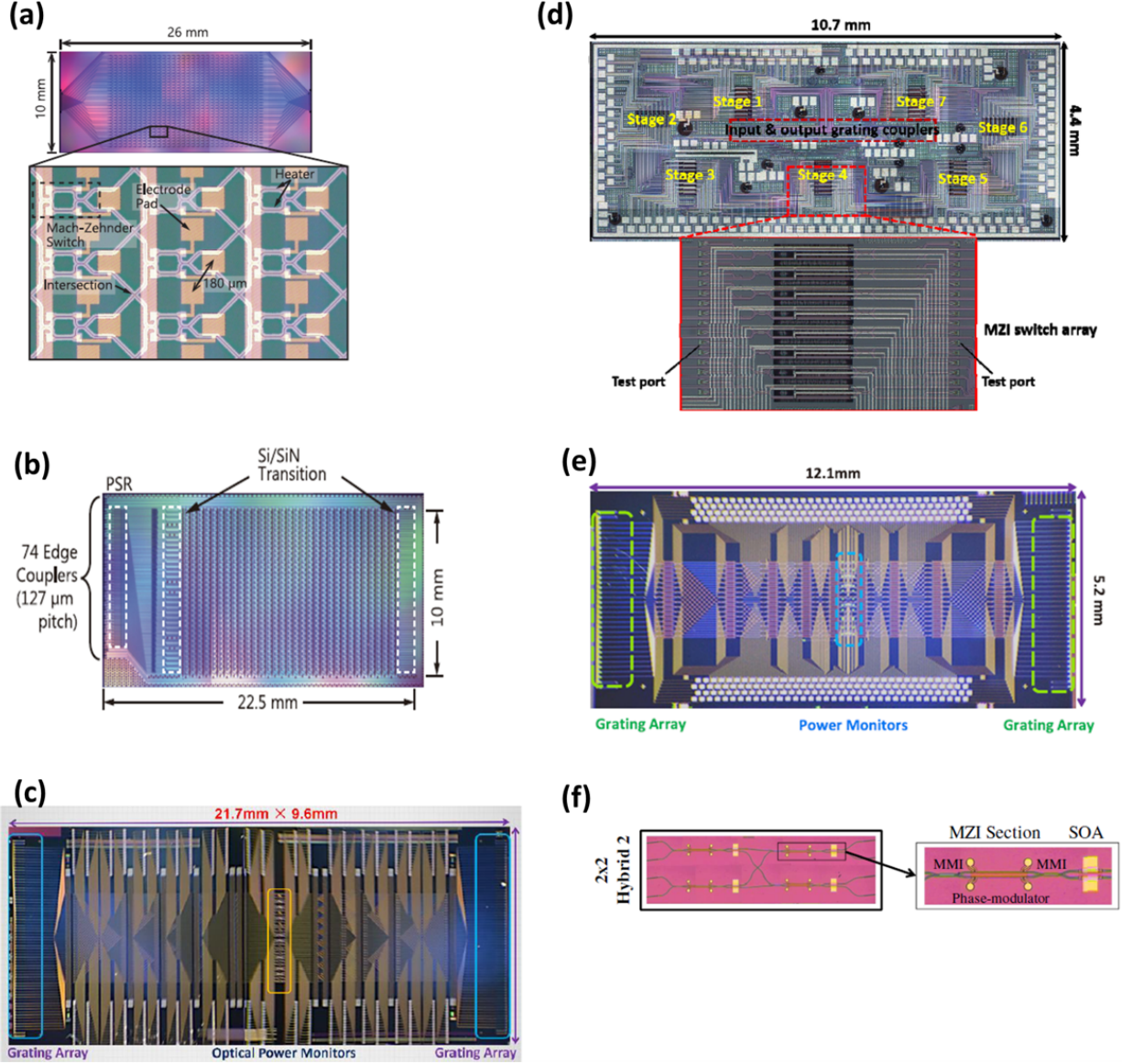


Figure 1.5: (a) Low insertion loss 32×32 MZI switch, reprinted from [47]. (b) Polarization-diversity 32×32 MZI switch, reprinted from [48]. (c) 64×64 T-O MZI switch, reprinted from [49]. (d) 16×16 E-O MZI switch, reprinted from [50]. (e) 32×32 E-O MZI switch, reprinted from [51]. (f) Hybrid 2×2 MZI-SOA module, reprinted from [52].

MZI switching circuits of 32×32 connectivity have been realized using thermo-optic (T-O) phase shifters with 6.1 dB on-chip loss [47]. To overcome the polarization dependency, a

polarization-diversity SiP MZI switch was further developed [48]. The current record for the T-O MZI switch is a 64×64 implementation in Bene topology [49]. For fast electro-optic (E-O) switching, carrier-injection based PIN junctions are employed. 16×16 and 32×32 E-O MZI-based switches were proposed in Refs. [50, 51]. Performance, however, can be limited due to the high insertion loss. Gain-integrated switches with semiconductor optical amplifiers (SOAs) for lossless operation can be applied to overcome this challenge [52]. Figure. 1.5 depicts the MZI based switches.

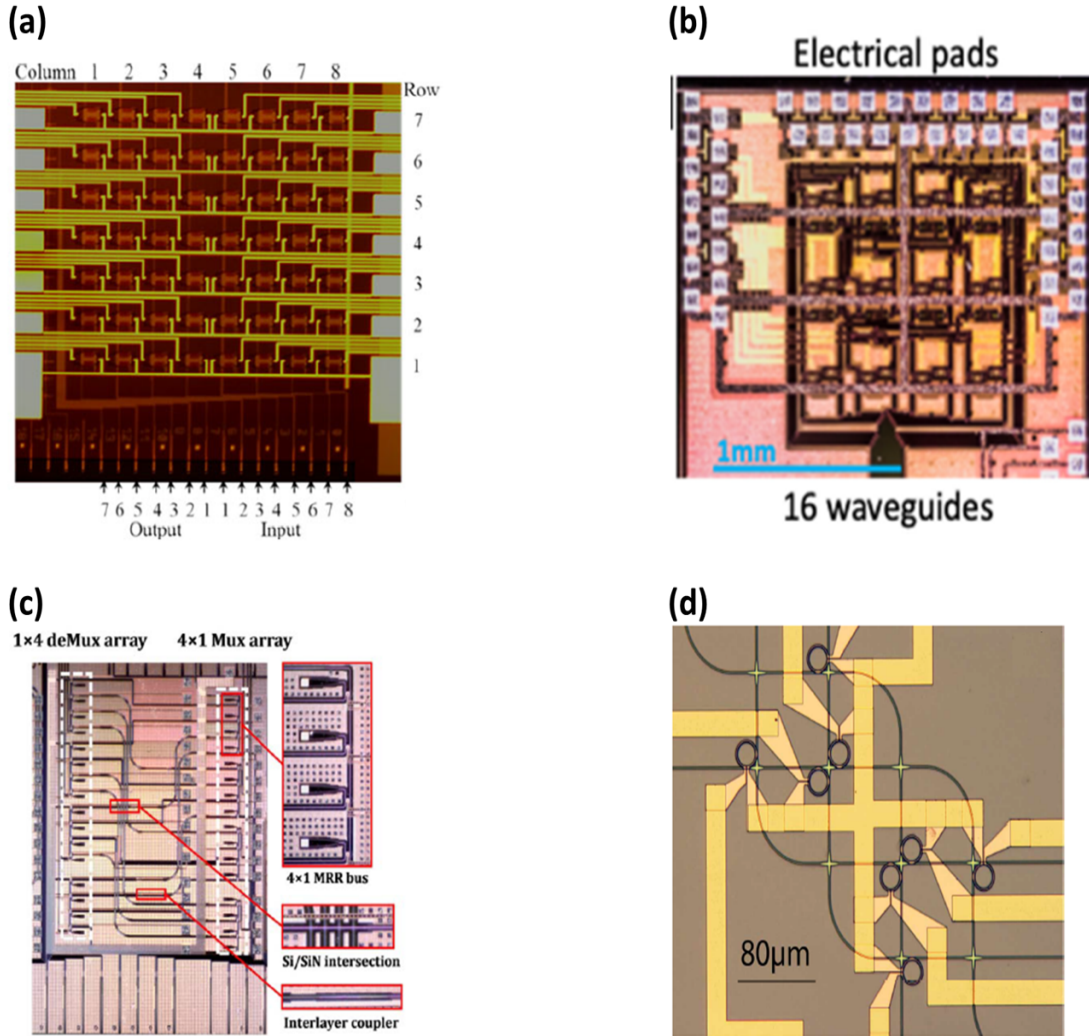


Figure 1.6: (a) 8×7 cross-bar switch, reprinted from [53]. (b) 8×8 Omega switch, reprinted from [54]. (c) 4×4 switch-and-select switch, reprinted from [55]. (d) 4×4 hitless switch, reprinted from [56].

MRR based devices show ultra-compact and energy-efficient potentials for optical switching. Recent work has demonstrated 8×7 cross-bar [53], 8×8 Omega [54], 4×4 switch-and-select [55], and 4×4 hitless [56] architectures. Add-drop filters assembled in a 1-D bus structure can act as spatial (de)multiplexers [57]. Thermal stabilization [58, 59] is necessary for MRR based switches to address wavelength drifts due to the thermal dependencies to the varying ambient temperature. Figure. 1.6 depicts the MRR based switches.

The largest-scale SiP switch fabric reported to date is the MEMS-actuated cross-bar switch with 240×240 connectivity, which consists of a 3×3 array of identical 80×80 switch blocks [60]. Maximum on-chip loss of 9.8 dB was reported. Multilayer bus waveguides can be used for eliminating waveguide crossings to reduce insertion loss and for addressing polarization sensitivity [61]. Recent work has shown successful fabrication of SiP MEMS using a commercial foundry with reduced driving voltage down to 9.45V [62].

More detailed discussions on the photonic switching technologies in datacenter/HPC systems can be found in the reviews [42, 43, 44]. We note that SiP switches are promising for optical switching in datacenter/HPC rack-to-rack applications; however, the loss should be further reduced before being deployed in practice. Approaches, such as (1) integration with SOAs, (2) improvement of coupling loss, and (3) progress on individual component to have a better loss performance, are being taken to further reduce the loss of silicon photonic switched architectures.

1.3 Silicon Photonic High Bandwidth Transceivers

The increasing mismatch between on-board and off-board bandwidth, as indicated in Fig. 1.4, calls for new interconnect design to overcome the network bottleneck and speed up applications such as the distributed training of extreme large models. Co-packaged silicon photonics with unprecedented bandwidth density, efficiency, and reach lends itself to the applications that require Tb/s data links. Direct silicon photonic connectivity for central processing units (CPUs), GPUs, FPGAs, domain-specific accelerators, and memory will impact a host of future deep learning applications.

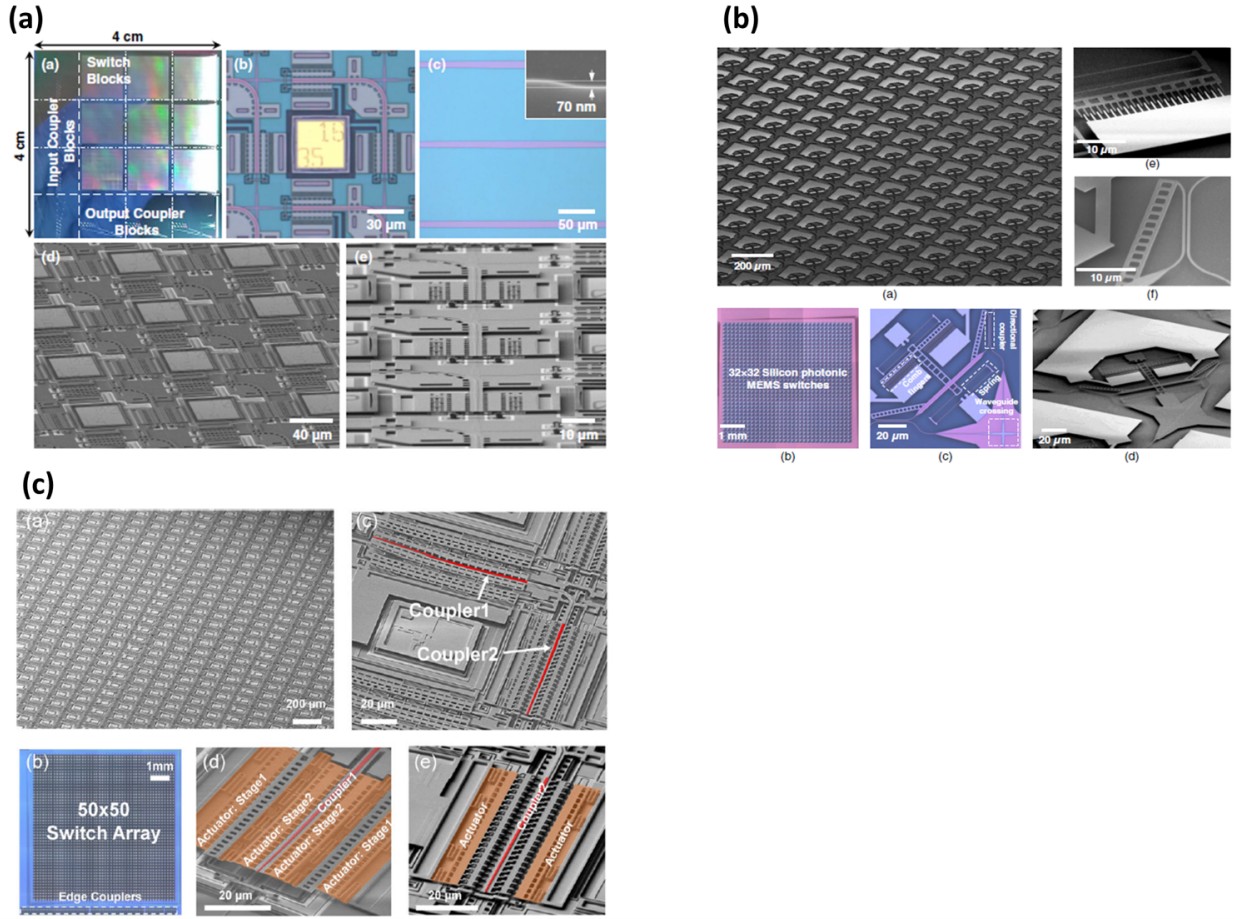


Figure 1.7: (a) 240×240 MEMS cross-bar switch, reprinted from [60]. (b) Polarization-Insensitive MEMS switch, reprinted from [61]. (c) 32×32 MEMS switch fabricated in a commercial foundry, reprinted from [62].

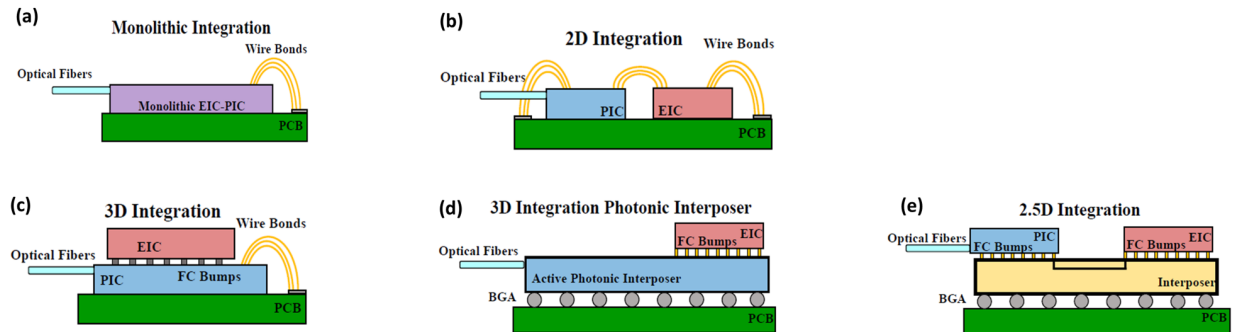


Figure 1.8: The various integration approaches to integrate PICs with EICs. (a) Monolithic integration. (b) 2D integration. (c) 3D integration. (d) 3D integration with active photonic interposer. (e) 2.5D integration. Reprinted from [63].

Figure 1.8 shows different methods of SiP co-packaging integration. In monolithic integration, both photonic integrated circuits (PICs) and electronic integrated circuits (EICs) components are fabricated in within the same die. It achieves minimum parasitics and simplified packaging with wire bonds for the connectivity to the PCB. However, it lags in cutting edge electronic performance and suffers from high waveguide loss, low photodiode responsivity, and low photodiode bandwidth. Figure 1.8(a) shows the monolithic integration approach. In 2D integration, the PIC and the EIC chips are placed side by side on a PCB, and wire bonds are used to connect the chips as well as to interface to the PCB. 2D integration allows most advanced technologies in photonics and electronics, but it shows limited I/O density, and introduces parasitic inductance. Figure 1.8(b) shows 2D integration approach.. Another approach is 3D integration, where EIC chip is flip chip (FC) bonded to the PIC chip. 3D integration increases the connectivity throughout the two-dimensional solder bumps, and reduces the parasitics between the EIC and PIC chips compared to 2D approach. However, PIC active components may be sensitive to the heat dissipated by the EIC chip. A promising approach is incorporate PIC components into an interposer. This is ideal for a dense, high performance I/O packaging; however, it is a relatively new technology and requires more development efforts. Figure 1.8 (c) and (d) shows the 3D integration and 3D active interposer integration approaches. A compromised solution is 2.5D integration, where PIC and EIC chips are flip chipped bonded to an interposer. The interposer supports EIC-to-PIC connections and fanouts necessary signals to PCB through a ball grid array (BGA). This approach also allows for high I/O bandwidth and density, and has fewer thermal concerns compared to the traditional 3D integration. Figure 1.8(e) shows the 2.5D integration approach. More detailed discussions can be found in Ref. [63].

The SiP transceiver architecture is the key enabler for high bandwidth interconnects. Coherent technology with advanced modulation format is widely used to provide high-capacity links in long-distance networks and metropolitan area networks. However, coherent systems are expensive for short-reach applications. They require narrow-linewidth lasers, power-consuming digital signal processing (DSP), and digital-to-analog and analog-to-digital (DA/AD) converters. Many schemes

have been proposed to reduce the power consumption, transceiver cost, and DSP complexity [64, 65, 66, 67].

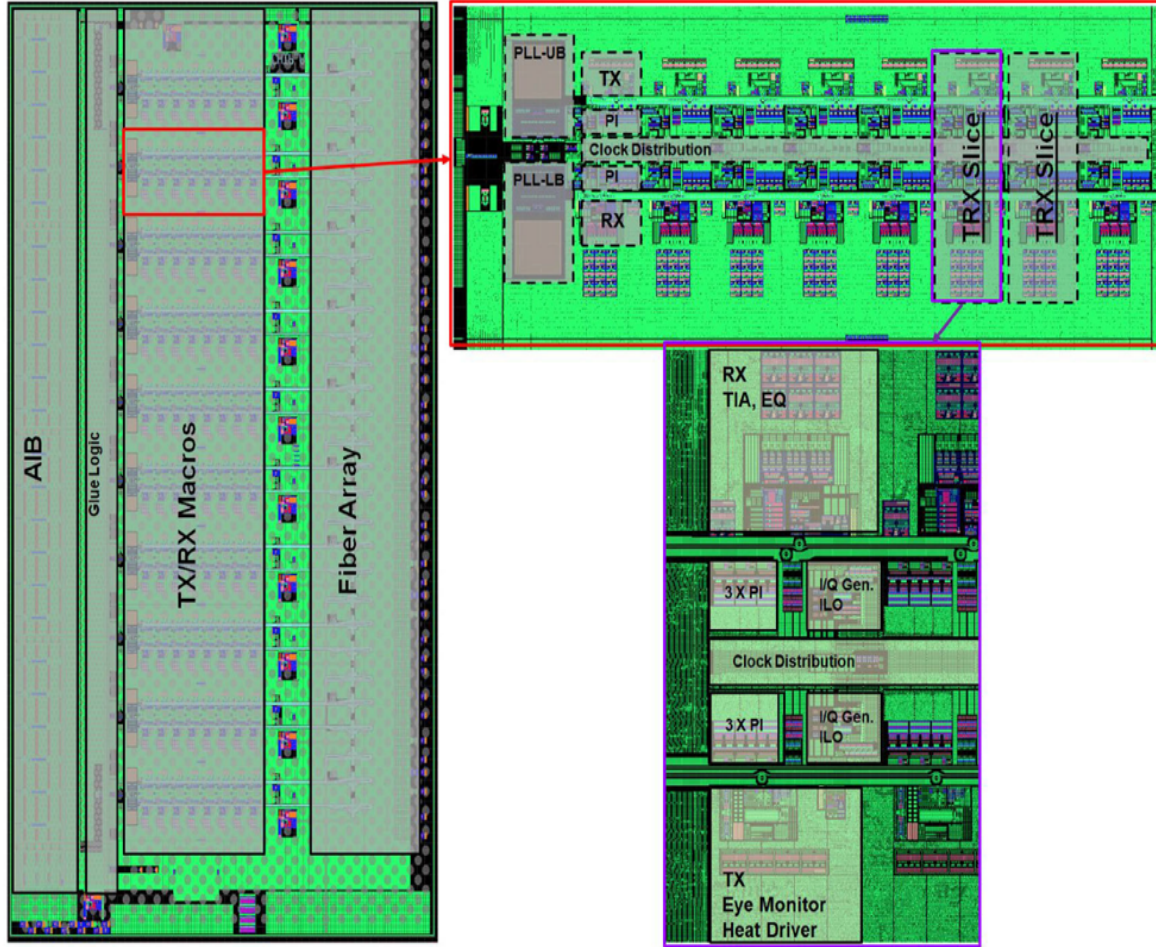


Figure 1.9: TeraPHY chiplet showing optical Tx/Rx macros, fiber array, and Tx/Rx circuits. Reprinted from [68].

Another approach is to still use simpler intensity-modulated direct-detection (IM-DD) links as preferred in today's DC applications. IM-DD links with on-off-keying (OOK) or pulse amplitude modulation 4-level (PAM 4) modulation formats can be reconstructed directly without forward error correction (FEC) for error-free operations. An example is TeraPHY [68] that supports up to 2Tbps bandwidth per chiplet. It uses a bus of 8 MRR based modulators running at 25Gbps each and a bus of 8 MRR based detectors form a transceiver macro. Each transceiver macro transmits and receives 8 different wavelengths of light. The chiplet then contains 10 macros to achieve the

Dense wavelength division multiplexing (DWDM) can be used to solve this issue and scale up high-bandwidth per fiber. The key challenges are the DWDM light source with hundreds of lines and a scalable transceiver architecture that is able to modulate and detect each of the wavelengths. Recently, a promising Kerr frequency comb-driven silicon photonic transceiver architecture [69] has been reported to achieve this goal. It leverages a frequency comb [70] that generates the DWDM light source. A de-interleaver is used to split the comb spectrum into subgroups. Each wavelength subgroup gets modulated by a bus of cascaded microdisk modulators, and then all the subgroups are recombined using an identical interleaver and coupled off the chip into a single fiber. At the receiver, the modulated comb lines are de-interleaved similarly and are incident on buses with cascaded MRR based detectors. Figure 1.10(a) shows the transceiver architecture and Fig.1.10(b) shows the transmitter chip. The Si₃N₄ dual-micro-resonator system for the comb generation is shown in Fig. 1.10(c).

Novel SiP transceiver technologies and proper packaging schemes have the potential to enable efficient multi-Tbps chip-to-chip communications and to meet the inter-chip bandwidth requirement posed by the scaling deep learning models.

1.4 Scope of Dissertation

The work in this dissertation addresses system level challenges posed by deep learning training, focuses on innovative architectural designs leveraging high performance silicon photonic interconnects, and demonstrates prototype systems to show the potential of integrating silicon photonic interconnects in future datacenters and HPCs. The chapters in this dissertation are comprised of the author's works in published peer reviewed journals and research conference proceedings.

In Chapter 2, we explore various optical spectral and spatial switching of SiP MRR based switches, such as unicast, multicast, and multiwavelength-select functionalities. We demonstrate our FPGA-based open-loop and closed-loop control over the switches. The successful control and the switching functionalities are essential for new architectural explorations in the following chapters.

In Chapter 3, we propose a photonic switched optically connected memory architecture for addressing memory constraints in deep learning. A “lite” (de)serialization scheme is used to reduce the communication overhead for memory transfers in optics. We evaluate the system performance on an experimental testbed with processing system, remote memory nodes, and SiP switches. The collective results show the potential of using optically connected memory for deep learning.

In Chapter 4, we demonstrate a silicon photonic switched architecture for deep learning in datacenter and HPC networks. The proposed architecture leverages reconfigurable optical switching fabrics to tackle the network challenges and accelerate distributed deep learning training. Experimental and simulation results suggest integrating SiP switches into datacenter and HPC systems can improve system performance and accelerate distributed deep learning training at scale.

In Chapter 5, we present two silicon photonic-enabled disaggregated system architectures for deep learning. The first architecture shows the concept of rack-scale disaggregation. The flexible optical switches enable resource disaggregation and defragmentation. The second architecture leverages comb laser sources and multi-wavelength selective MRR-based cross-bar switches to enable the all-to-all high bandwidth communication. We observe improved system performance of our proposed architecture compared to other existing topologies.

Lastly, in Chapter 6, we report our progress on the development of optical network interface cards for the FLEET project. We show the design and packaging flow to expand PCIe communication into optical domain. Test packages for system development and verification are presented. This chapter shows integrating of silicon photonic technologies into high performance systems.

Chapter 2: FPGA-controlled Silicon Photonic Interconnects

2.1 Introduction

Reconfigurable networks and optical circuit switches (OCSs) based architectures have been proposed to be deployed in datacenter and HPC systems. These architectures including Helios [71], Mordia [72], ProjecToR [73], Flexfly [74], and many others, can establish high-bandwidth connections to optimize network performance for frequently-communicating nodes. Commercially available technologies [38, 39, 40] suffer from high cost per port due to their rigorous calibration and installation of discrete components. For low-cost adoption, lithography-based photonic integration technologies hold great promise for large-scale optical integrated switch fabrics by reducing the device footprint and also the overhead in terms of assembly and calibration. Planar integrated optical switches have been developed on several material platforms, such as indium phosphide, lithium niobate, silica, and silicon [52, 75, 76, 55].

Silicon photonics, fabricated in high volume CMOS compatible foundries, is promising for low-cost, power-efficient interconnects. The primary switching cells that are being explored are MZIs [47], MRRs [54], and MEMS-actuated couplers [60]. A basic 1×2 MZI switching cell is shown in Fig. 2.1(a). The phase delays in the two arms can be changed by their corresponding phase shifters through electronic bias input to thermo-optic heaters or electro-optic drivers for carrier injection/depletion. The resultant phase differences between the MZI arms can direct the input light to output port #1 (bar state) or output port #2 (cross state), respectively. Same bias control mechanism can be applied to MRRs. A fundamental 1×2 MRR switching cell is shown in Fig. 2.1(b). The input light can be directed to output port #1 (through state) or output port #2 (drop state), respectively. The difference is that MZI switches are broadband switches, while MRR switches are narrowband devices and only periodic wavelengths with a certain free-spectrum-range

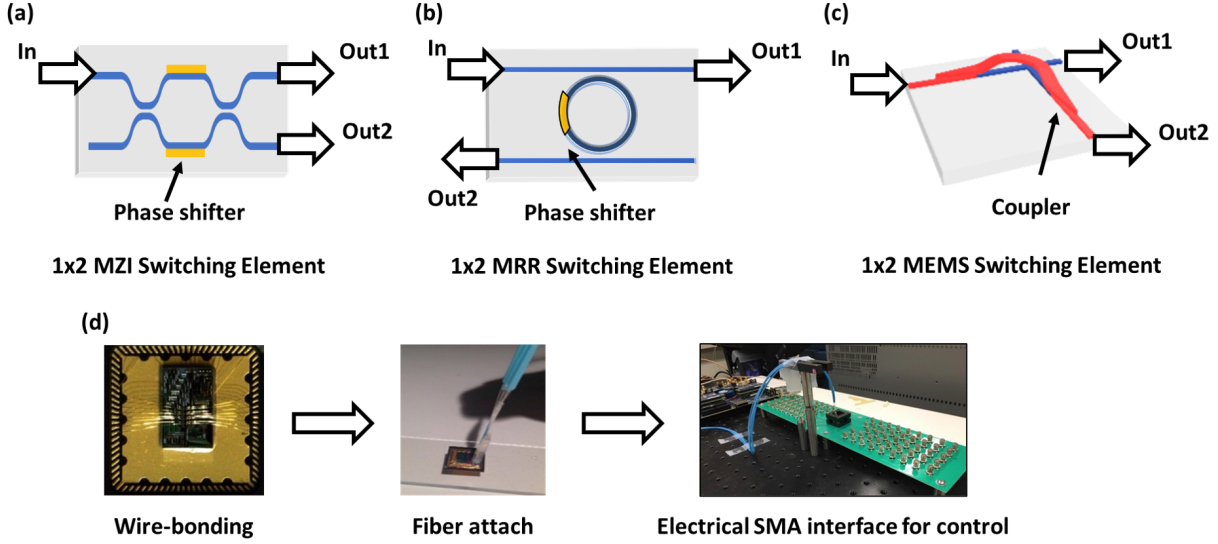


Figure 2.1: (a) 1×2 MZI cell. (b) 1×2 MRR cell. (c) 1×2 MEMS-actuated coupler cell. (d) Packaging with wirebonding and fiber attachment

(FSR) can be dropped. Figure 2.1(c) the fundamental 1×2 MEMS switch. By biasing the electrodes of MEMS, the input light can be forwarded to output port #1 (off state) or redirected through the adiabatic couplers to output port #2 (on state), respectively. In general, all three switching cells can be tuned by electrical bias voltages. Figure 2.1(d) shows a generic packaging scheme for the interface to a large radix switch chip that consists of many of those basic switching cells. Since the tuning scheme is similar, only the MRR-based switches are mainly discussed in the rest of this chapter.

2.2 Optical Switching in Cascaded-MRR Switch

Figure 2.2(a) shows an array of eight cascaded SiP MRRs. This cascaded MRR switch can achieve various optical functionalities such as unicast, multicast, and multiwavelength-select. Figure 2.2(b) illustrates the responses of the MRRs in the chip when no bias voltage is applied. By design, the responses of the rings are separated by 1.27 nm (160 GHz) with an FSR of 13 nm and 3 dB bandwidth of 0.7 nm.

Figure 2.2(c) shows a unicast operation where the input data on the wavelength denoted by a red

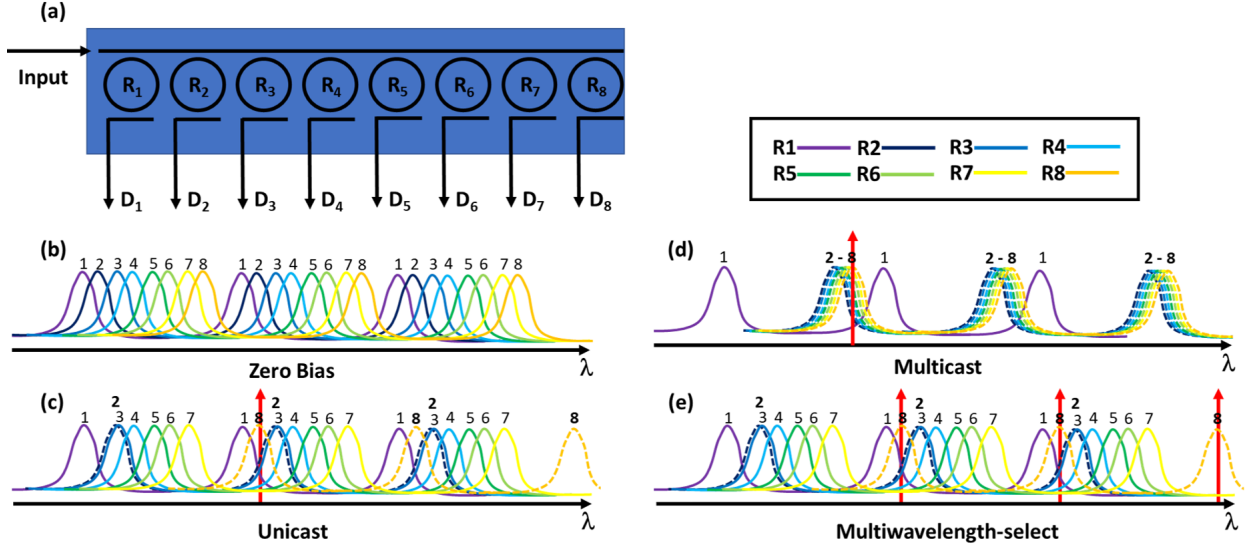


Figure 2.2: (a) Switch architecture with 8 Cascaded MRRs. (b) Zero bias scenario. (c) Unicast scenario. (d) Multicast scenario. (e) Multiwavelength-select scenario.

vertical arrow is routed to output port 8 (dashed black curve). To obtain this mode of operation, i) the resonance of ring R₂ must be detuned to prevent dropping at port 2 because R₂ has the precedence over R₈ in the MRR array, and ii) the resonance of ring R₈ must be tuned to the red wavelength. Detuning is required for ring R₂ to achieve error free operation for ring R₈. As the bias voltage over R₂ is increased gradually, its resonance is shifted to allow the signal to propagate to R₈. The bit-error-rate (BER) is needed to verify the amount of detuning necessary in order to reach the error free transmission (BER = 10e-12) with negligible crosstalk effects [77]. At a bias of 0.85 V the signal is fully routed to the output port 8. We estimated that this amount of detuning is about 1.08 nm which corresponds to a channel suppression of 10 dB between the desired output port (R₈) and the detuned MRR (R₂). This amount of detuning is chosen for power efficiency; any further detuning will cost more energy-per-bit while introducing negligible improvement on the BER of the routed signals.

Figure 2.2(d) illustrates a one-to-seven multicasting operation. This operation is possible by aligning the Lorentzian response of each MRR so that the power of the optical signal is divided equally among the desired output ports; i.e. tuning the rings to the appropriate resonances, starting

from the last MRR participating in the multicast operation. The last MRR, R8 in this example, is tuned so that its resonance aligns exactly with the input wavelength, allowing maximal transmission over its drop path. R7 is then tuned to its 3dB bandwidth point, allowing a drop of 50% of the optical power. Continuing with this approach R6, R5, R4, R3 and R2 are tuned to the following drop power ratios: 33.33%, 25%, 20%, 16.67% and 14.28%, respectively, i.e. following a harmonic series ($1, 1/2, 1/3, 1/4 \dots$). When the process is complete, each of these seven rings will equally drop 14.28% of input optical power coupled into the SiP chip. Realizing multicast exhibits the fine tuning levels, and demonstrates the potential of SiP for error-free multicast operation of one stream of data over a single wavelength.

Figure 2.2(e) shows a three-to-one multiwavelength-select operation. This operation leverages the ring's FSR. Each ring is capable of dropping a train of input signals as long as their input wavelengths are separated from each other with a distance as the ring's FSR. Similar to the unicast case, the resonance of ring R8 is tuned to each of the input red wavelengths, and ring R2's resonance is detuned to prevent dropping them at port 2. This operation allows to aggregate more bandwidth with multiple wavelengths at drop port of each ring, and potentially enables a shuffling operation in a cross-bar ring-based switch fabric. More discussions on the cross-bar switch design and its usage can be found in chapter 7.

2.3 FPGA-Based Open Loop Control

2.3.1 System Architecture

Figure 1 shows a block diagram of our proposed system architecture. The main components are fast tunable lasers (TL), a SiP chip, a field programmable gate arrays (FPGA) and a computer that provides an interface to the user. Depending on the desired configuration, the computer sends operation messages to the FPGA via a serial data transfer interface. The TLs can adjust its output optical power and the wavelength of operation over the C-band. The FPGA controls the array of eight cascaded MRRs and sets their resonances. As shown in Fig. 1, an external modulator is used to encode data onto the laser outputs. Altogether, the system is capable routing of optical data to

any of the eight output ports, splitting the incoming data to a number of output ports, as well as selecting/forwarding multiple wavelengths to a single MRR.

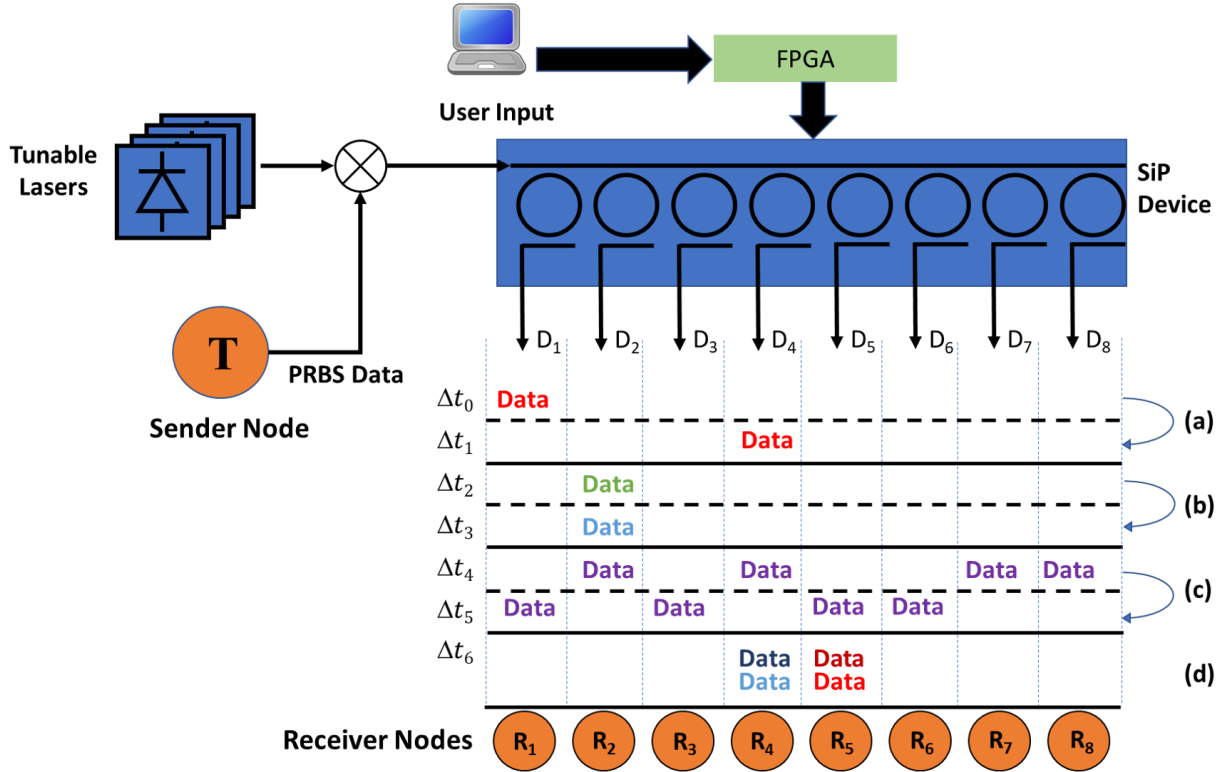


Figure 2.3: Open-loop control scheme.

The time-slot diagram in Fig. 2.3 illustrates four possible operations of the system. In this example, the different colors of the data in the figure correspond to different wavelengths. During slot Δt_0 the data stream is routed to output R1. During Δt_1 slot the data stream is switched to output R4 (Fig. 1(a)) without changing the operating wavelength (red color). In the second case, shown in Fig. 1 (b), wavelength routing is performed. During time slot Δt_2 the data is transmitted to output R2 on a carrying wavelength λ_1 (green color) while in slot Δt_3 the output port stays the same but the wavelength changes to λ_2 (blue color). Multicast (one to many) operations are also performed. During Δt_4 , an incoming data stream modulated on the wavelength (purple color) is split among four output ports: R2, R4, R7, and R8. In the following time slot Δt_5 , the configuration is modified to R1, R3, R5, and R6. Finally, during time slot Δt_6 , R2 and R3 are used to perform

multiwavelength-select operations. Wavelengths in blue and dark blue separated a FSR are dropped by R4. Similarly, the adjacent R5 drops wavelengths in red and dark red. The minimum transition time between the time slots is determined by the reconfiguration time of the SiP device from the moment the FPGA apply the bias voltages for new configurations.

The operation of the system is not limited to the particular sequential illustration presented in Fig. 2.3. The SiP device can perform unicast (one to one), multicast (one to many), and broadcast (one to all), multiwavelength-select (many to one) operations, numerous deterministic combinations are possible without any constraints.

2.3.2 Experimental Demonstration for Unicast and Multicast

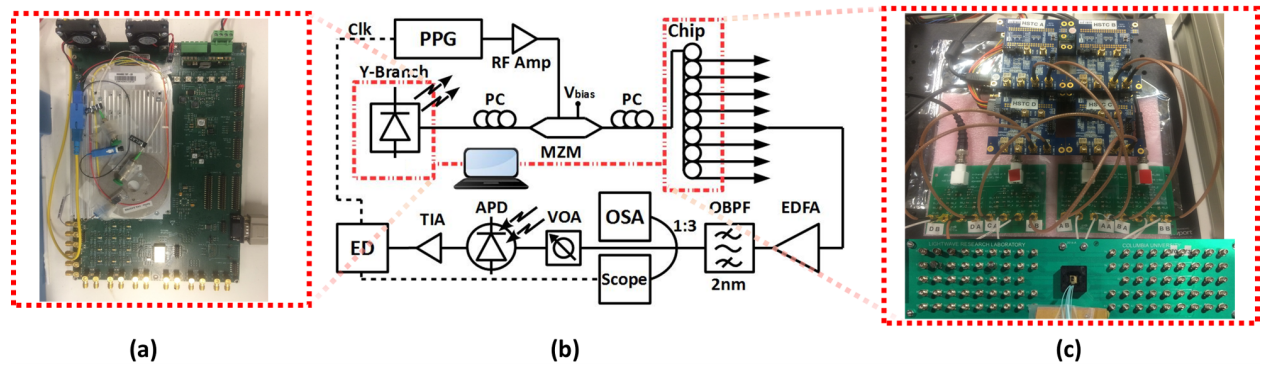


Figure 2.4: (a) Tunable laser. (b) Schematic of the unicast and multicast experimental setup. (c) FPGA-based switch controller and the packaged MRR switch on a PCB.

The schematic of the experimental setup is shown in Fig. 2.4. A tunable Y-branch laser [78] (Fig. 2.4(a)) was used to output light at various frequencies across the ITU 100 GHz C-band grid. The packaged SiP device along with its control plane is shown in Fig. 2.4(c). The control plane is based on an Stratix III EP3SL150 FPGA capable of hosting and controlling eight parallel digital-to-analog converters (65 MHz DAC) with 14 bits of resolution. The output voltage from the DACs (0 – 1V) is amplified by 5 in the gain stage to achieve a full FSR swing for each MRR. The amplified control signals are connected to a break-out printed-circuit-board (PCB) which hosts the SiP chip. After fabrication, the chip was attached to, and wire-bonded on, a standard electrical IC

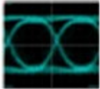
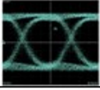





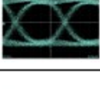








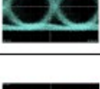

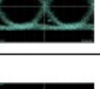










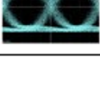

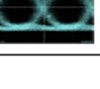




	Drop port 1	Drop port 2	Drop port 3	Drop port 4	Drop port 5	Drop port 6	Drop port 7	Drop port 8
Unicast								
Multicast of 2								
Multicast of 3								
Multicast of 4								
Multicast of 5								
Multicast of 6								
Multicast of 7								
Broadcast								

Figure 2.5: Experimental results of 8 different cases including unicast, multicast, and broadcast.

package as shown Fig. 2.1(d). The silicon photonic chip used in this work is not equipped with any temperature sensors, so no active temperature stabilization procedure is included in the design of the chip. However, the chip is sitting on a heat sink in the IC package that passively regulates its temperature to the ambient temperature level.

To test various switching scenarios, a programmable pulse generator (PPG) was used to output a 10 Gbps PRBS ($2^{31}-1$) signal, which was then electrically amplified before being modulated onto the optical carrier using a Mach-Zehnder modulator. The optical signal was then sent to the SiP device, configured through the software control plane, to route the signal to the desired output port(s). An erbium-doped fiber amplifier (EDFA) was required to amplify the received signal to

compensate coupling loss of the chip. Using a 1:3 divider, the optical signal was directed to an OSA, an oscilloscope to observe the received eye diagram and an APD. The power falling on the APD was kept at a constant level of -8 dBm to avoid saturation. The output of the photo-receiver was sent to the bit error rate tester (BERT) for BER measurement. The BERT and oscilloscope were both triggered with the same clock signal from the PPG.

Figure 2.5 shows experimental measurements for multicast and broadcast operations. All the results were set to perform at 1548.11 nm. First, the control plane was set to route the data to R2. Then the SiP device was reconfigured to multicast the data to ports R2 and R7. The BER and the eye-diagrams were captured at the two output ports separately to verify error-free operation. This process continued to higher multicast port counts until it covered all the eight ports (broadcast). The tested multicast cases are marked by eye-diagram results in each row of Fig. 2.5.

2.3.3 Experimental Demonstration for Multiwavelength-select

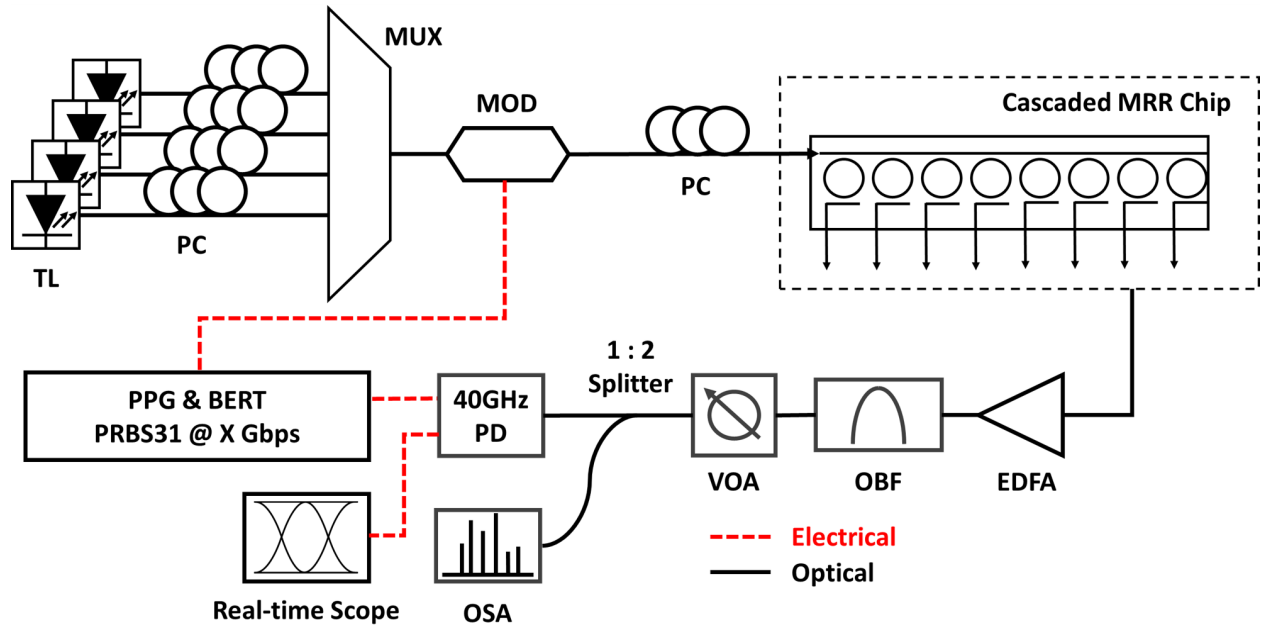


Figure 2.6: Schematic of the multiwavelength-select experimental setup.

The schematic of the experimental setup is shown in Fig. 2.6. Four TLs are used to output four

wavelengths. First two wavelengths are located at the resonances of R1 and the third and fourth wavelengths are at R2's resonances. An optical multiplexer (MUX) is used to combine them into a WDM channel. PCs maximize the optical power going into a broadband modulator (MOD) as well as the chip. A PPG generates the pseudo random bit sequence (PRBS) signal and drives the MOD at 10 Gb/s/ λ . R1 is tuned to drops two wavelengths distant from a FSR, and R2 is tuned to drop the other two. Using a 1:2 splitter, the WDM signal dropped by each ring was directed to an OSA to visualize the spectrum, and to a 40GHz photo-detector (PD) at the same time. In between, an EDFA is used to compensate the coupling loss of the chip, and an optical bandpass filter (OBF) is used for only passing the corresponding wavelength for each measurement. A variable optical attenuator (VOA) reduces the input power if necessary to protect the PD. Two high-speed electrical outputs are available from the PD, and the outputs are connected to a BERT for the BER measurement, and to a real-time scope for plotting the eye diagrams.

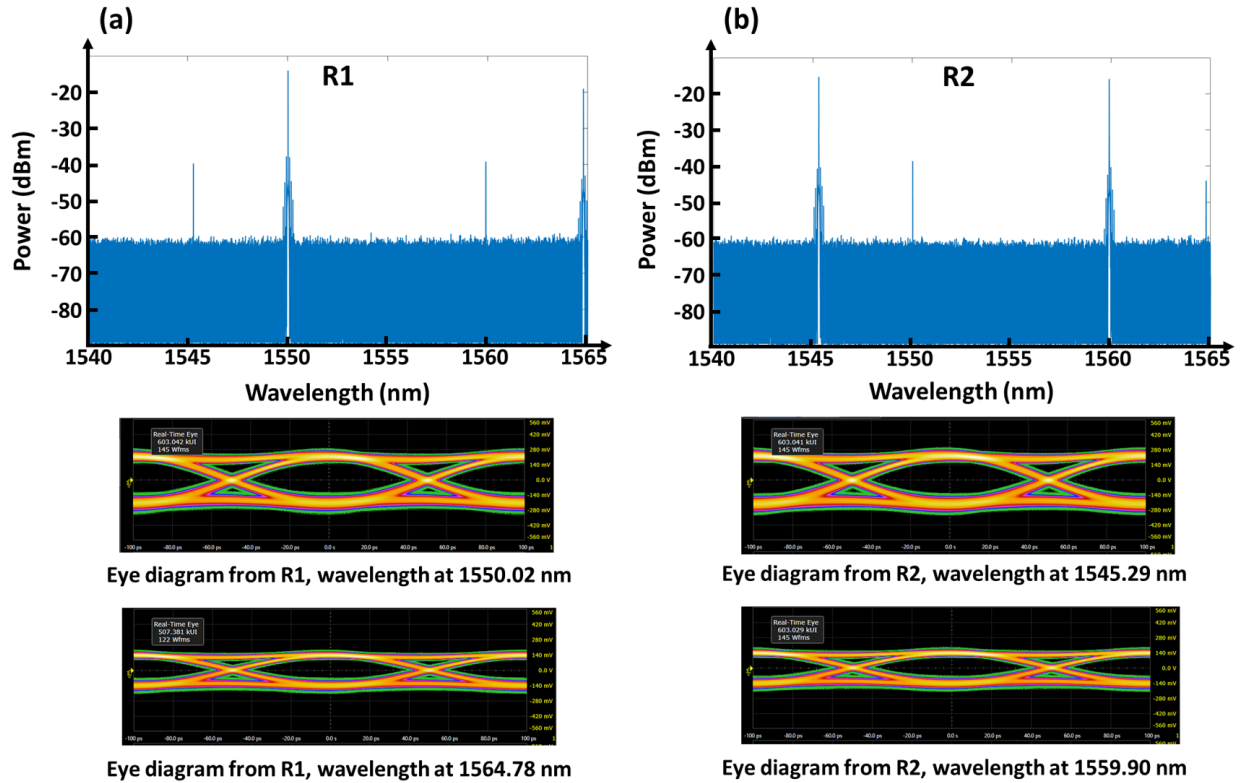


Figure 2.7: Experimental results for multiwavelength-select operations.

Figure 2.7 shows experimental measurements for multiwavelength-select operations. There are four modulated signals at different wavelengths. The bias voltage was set to route the data signals at 1545.29 nm and 1559.90 nm to R2, and R1 was tuned to drop the data signals at 1550.02 nm and 1564.78 nm. The optical spectrum shows the modulated signals received at the SOA for R1 and R2 in Fig. 2.7(a) and (b), respectively. The eye-diagrams of each wavelength, also shown in Fig. 2.7(a) and (b), were captured at the two output ports separately to verify error-free operation. This process shows the potential of MRR to aggregate more bandwidth by leveraging this multiwavelength-select functionality.

2.4 FPGA-Based Closed Loop Control

As Silicon Photonic technology matures, fabrication foundries are able to integrate large scale optical components on a single chip [79]. The foundries support a library of components including Mach-Zehnders modulators (MZM), spatial switches, optical splitters and photo-detectors. Among these, the MRR is one of the most versatile optical building blocks with a narrow wavelength selectivity. Along with modulation capabilities, the MRR allows wavelength filtering, hence applicable for (de)multiplexing the WDM signals at a small footprint and low energy consumption. However, MRR resonance characteristics make it susceptible to deviation from the optimal operations. Fabrication and thermal variations can shift the resonance and cause high optical losses. A common method to address these problems is an integration of a micro-heater next to the MRR to tune the local temperature and fix resonance to the desired wavelength. An analog feedback loop [58] can tap a portion of the optical signal dropped from the MRR and adjust the bias on the heater to maximize the transmission power. However, a digital interface is still needed for a selective wavelength operation. While Digital feedback solutions [80, 81, 82] can achieve both functionalities, they rely on traditional digital-to-analog (DACs) and analog-to-digital converters (ADCs), which require a large number of parallel input/output (I/O) pins. For example, a 4-channel receiver [80] with 8-bit DACs and 16-bit ADCs requires 24 I/O pins for each MRR and 96 in total. Reference [81] successfully decreased the number of DACs and ADCs in a cascaded MRR structure, however tens of

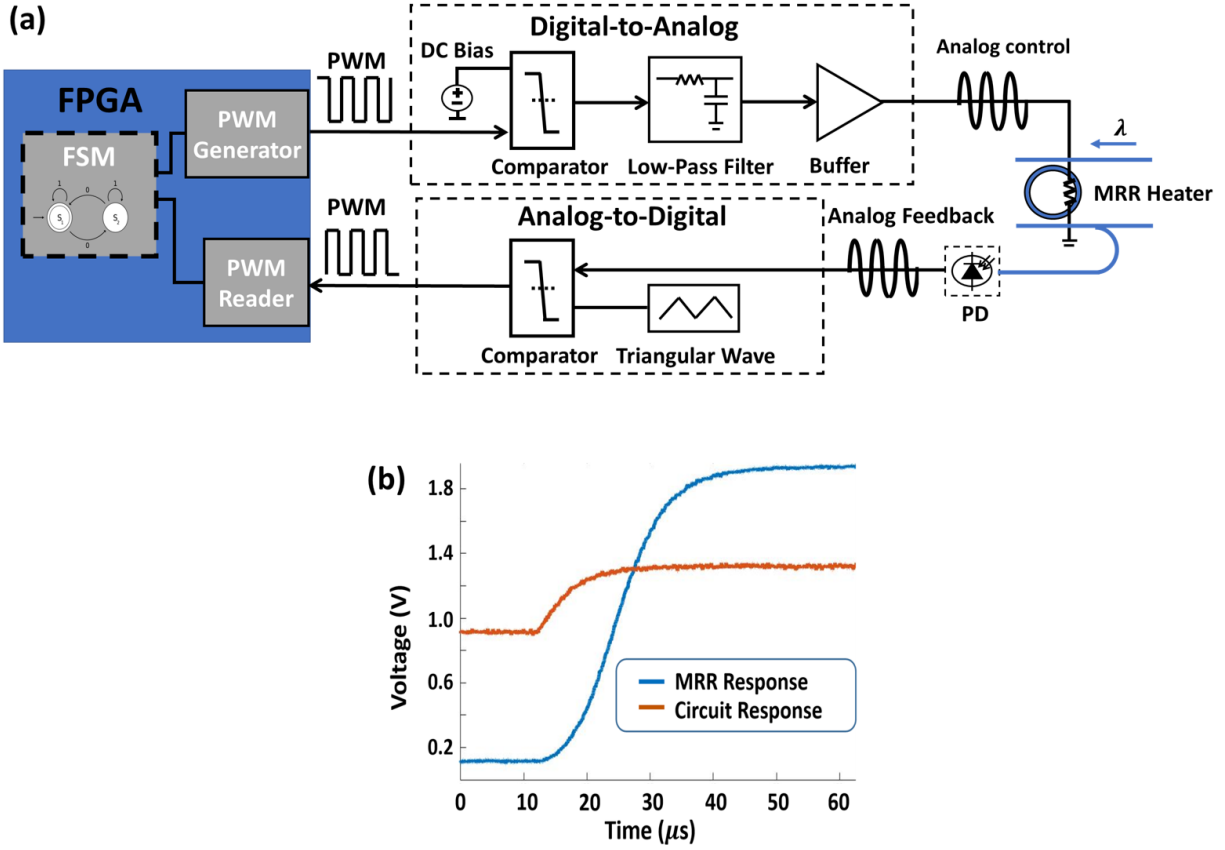


Figure 2.8: (a) Schematic of digital-to-analog and analog-to-digital control and feedback (b) Transient of digital-to-analog circuit response and MRR response due to a changed duty cycle of the PWM control signal.

I/O pins are still required.

2.4.1 Closed-loop Single-wire DAC and ADC Architecture

Figure 2.8(a) shows the building blocks for realizing the wavelength switching and feedback for reliable operations of an MRR. On the control side, a PWM control signal is generated by the FPGA and converted to bias voltage to tune the resonance of the MRR. On the feedback side, the analog feedback signal coming from the MRR is also converted to a PWM signal for the FPGA to process. Based on the input and the output PWM signals, the feedback control algorithm implemented as a finite-state-machine (FSM) will tune the resonance of the MRR for optimized operations.

The digital-to-analog converter is implemented for the control side. By increasing/decreasing the duty cycle of the PWM control signal, the bias voltage on the heater is increased/decreased. The PWM generator first generates the signal at 1.2MHz, which passes through the converter consisting of a comparator, a low-pass filter and a buffer for cleaning, converting and supplying enough current to the MRR heater. Fig. 2.8(b) shows the circuit and MRR response time are 6.20 μ s and 30.3 μ s due to a duty cycle change from 29.67% to 42.26%. The time response of the circuit is 5 times faster than the MRR, which guarantees a fast tuning procedure. The clock frequency of the FPGA at 625MHz provides a 9-bit resolution (625M divided by 1.2MHz) with a single wire.

For the feedback, a portion of the optical signal is tapped from the MRR and converted to an analog signal at the photo-detector (PD). The signal then passes through the analog-to-digital converter consisting of a comparator and a triangular wave generator. The analog signal from the PD is compared with the triangular wave at 150 KHz, resulting in a feedback PWM signal, received by the reader, also at 150 KHz. The resolution of our analog-to-digital converter is 12-bit (625MHz divided by 150KHz), with only a single wire required.

The single-wire ADC/DAC structure is also compatible with the feedback control algorithm for tuning the MRR to maximize transmission power in unicast and divide power equally in multicast operations. The algorithm starts with an estimate of the PWM control signal that sets the bias voltage for each required operation. However, due to thermal fluctuations, the resonance may shift and need to be fixed to the desired wavelength. In order to achieve this, the algorithm first detects the sign of gradient [82] at the estimated duty cycle of applied PWM signal, and tunes the resonance based on expected operations as well. For unicast, the algorithm keeps increasing/decreasing the duty cycle in steps until the maximized transmission power is achieved. For multicast, the algorithm begins with the last participating MRR to maximize the dropped power, and all other participating MRRs in the cascaded structure waits until the preceding MRR is finished. They are then tuned to drop optical power that is equal to the first tuned MRR. The transient operation of the feedback tuning procedures is presented in the results section.

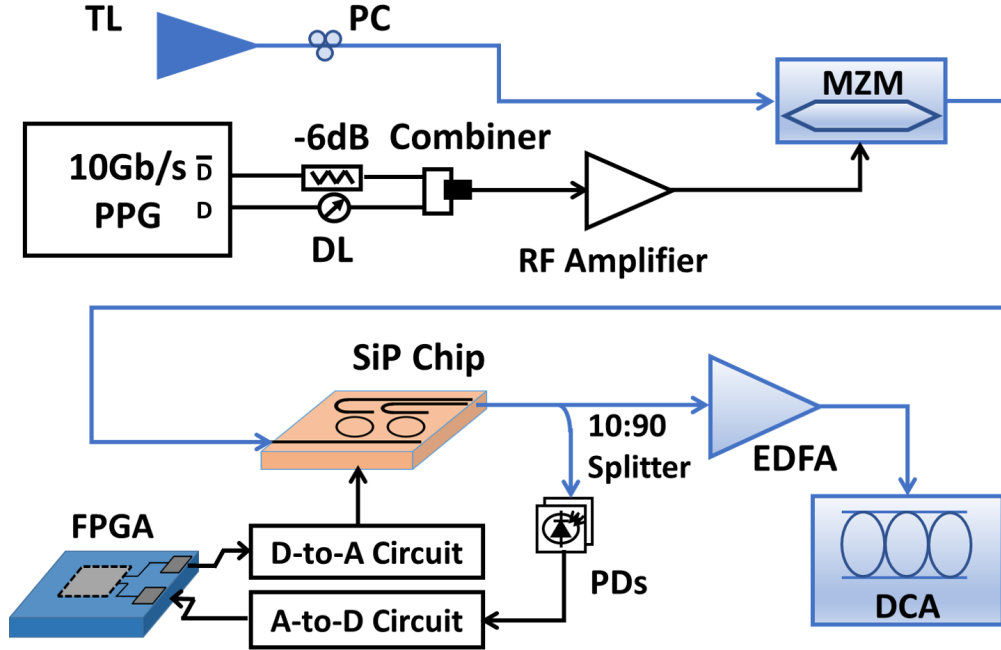


Figure 2.9: Experimental setup demonstrating the single-wire ADC and DAC silicon photonic circuits evaluated with PAM-4 signal.

2.4.2 Experimental Setup

The experimental setup is shown in Fig. 2.9. The proposed control and feedback architecture is evaluated with two cascaded MRRs for unicast and multicast. A TL is set to a specific wavelength on the C-band. A 10Gb/s PRBS is generated using a PPG. The positive output data (D) is attenuated by 6dB and negative data (\bar{D}) output is phase-matched with an electrical delay line (ED). Subsequently, D and (\bar{D}) are combined by a combiner and amplified using an RF amplifier to drive the MZM to generate the 20Gb/s PAM-4 signal. The polarization controller (PC) is used to set the maximized optical signal to the SiP Chip.

An FPGA configures each MRR with bias voltage through the single-wire DAC and ADC. Two additional I/O pins from the FPGA are used in multicast operation. An optical splitter is used to tap 10% of the optical power to a PD feeding it back to the FPGA at each output port of the MRRs. The other 90% is amplified by an erbium doped fiber amplifier (EDFA) to compensate for the 15dB fiber to grating coupler loss. The optical PAM-4 signal is directed to a digital communication

analyzer (DCA) for capturing the eye-diagram and estimating the sample-error rate (SER).

2.4.3 Results

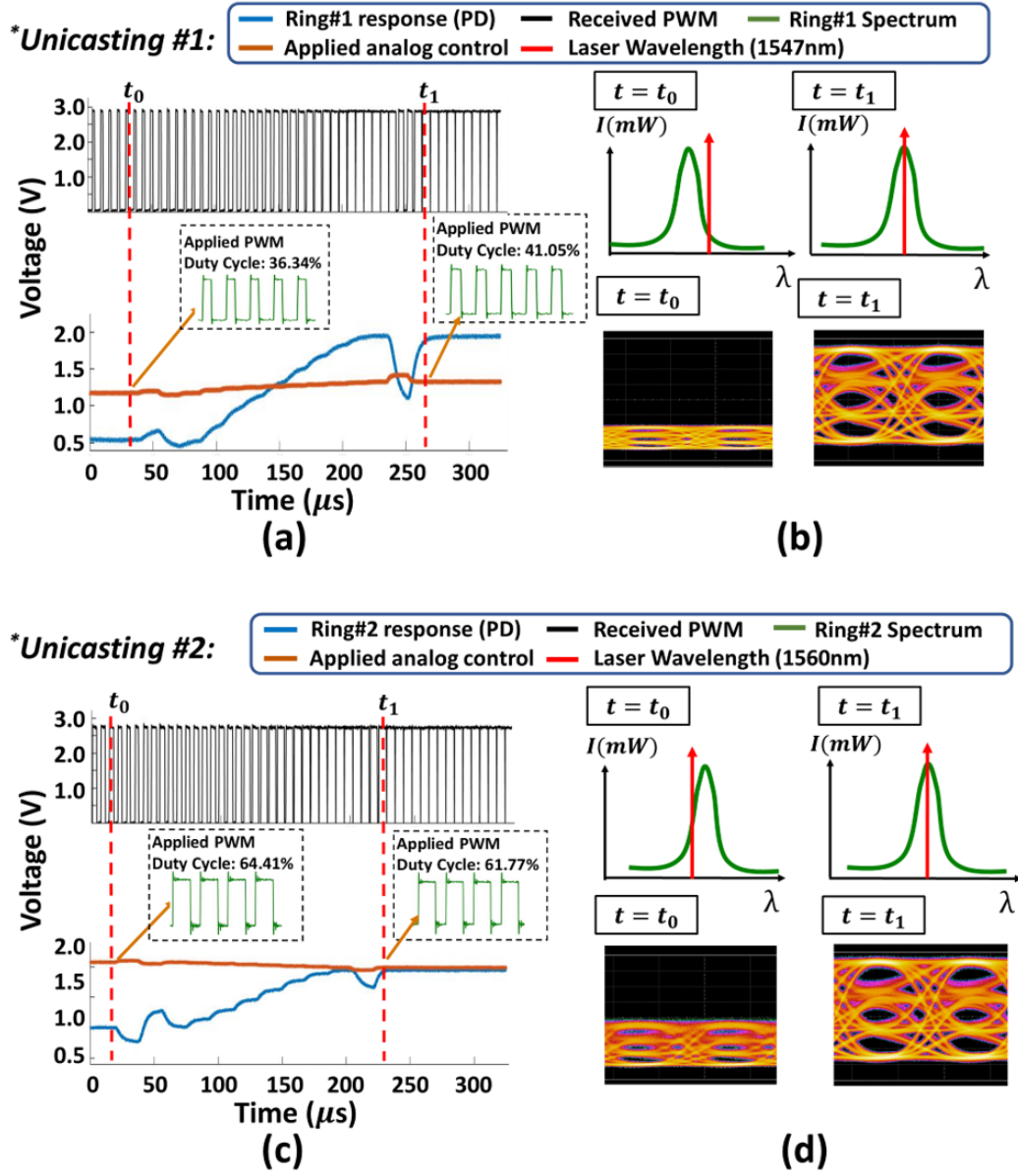


Figure 2.10: (a) Transient responses for unicasting of MRR #1. (b) Corresponding spectrums and eye diagrams of MRR #1 before and after tuning. (c) Transient responses for unicasting of MRR #2. (d) Corresponding spectrums and eye diagrams of MRR #2 before and after tuning.

Figure 2.10 shows the experimental results of two unicast cases. The resonance of MRR #1 is 6dB deviated from the input wavelength at 1547nm (Fig. 2.10(a)). An increased duty cycle of

applied PWM signal is required. The optical transient response (blue curve) is captured in the PD and the feedback process starts at $t = 37 \mu s$ (t_0) and ends at $t = 271 \mu s$ (t_1). The PWM signal received by the FPGA is shown in black, where the duty cycle gradually increases from 27.24% at $t = t_0$ until the feedback algorithm detects a negative change and returns to the maximum point where the duty cycle stays at 92.78%. The orange curve shows the applied voltage on the MRR heater during the tuning operation. The applied PWM control signal and its duty cycle are shown in the black box. Fig. 2.10(b) top illustrates the resonance of MRR #1 versus the input wavelength before and after tuning. Eye diagrams are also shown at the bottom. The corresponding SERs are $2e-6$ and $7e-18$. The results for MRR #2 are collected with input wavelength at 1560nm are shown in Fig. 2.10(c) and (d). In this scenario, a decreased PWM duty cycle is needed. The tuning procedure takes $218 \mu s$ in total and achieves 3dB increased optical power. The corresponding received PWM duty cycles are 50.26% at the beginning and 96.20% at the end. The applied PWM's duty cycle decreases from 64.41% to 61.77% during tuning. MRR #2's resonance versus the input wavelength is shown at the top of Fig. 2.10(d). The resulting eye diagrams are shown in the bottom and SERs are $2e-7$ and $3e-18$.

Figure 2.11 shows the experimental results for the multicasting data at 1547 nm through both MRRs simultaneously. The feedback operation as captured in the PDs is shown in Fig. (a) (blue and orange curve). The feedback starting at $t = 17 \mu s$ (t_0) is completed in $327 \mu s$. At $t = 228 \mu s$ (t_1), the resonance of MRR #2 reaches the input wavelength where maximum output power is obtained and the algorithm starts to tune MRR #1. Multicast is achieved as MRR #1 drops the same optical power as MRR #2 at $t = 344 \mu s$ (t_2). The applied PWM and its duty cycle for both MRRs is labeled in the black box. Fig. 2.11(b) shows the transient of received PWM signals. During the tuning procedure, the duty cycle of PWM received signal from MRR #2 (in the top of Fig. 2.11(b)) is 27.24%, 71.01% and 49.21% at t_0 , t_1 , and t_2 . The transient of the received PWM signal from MRR #1 is shown in the bottom of Fig. 2.11(b), with the duty cycle increasing from 29.28% at t_1 to 50.77% at t_2 . The resonances of MRR #1 and MRR #2 are shown in the green and purple curve before tuning and the eye diagrams for detuned MRR #1 and MRR #2 are shown in the bottom

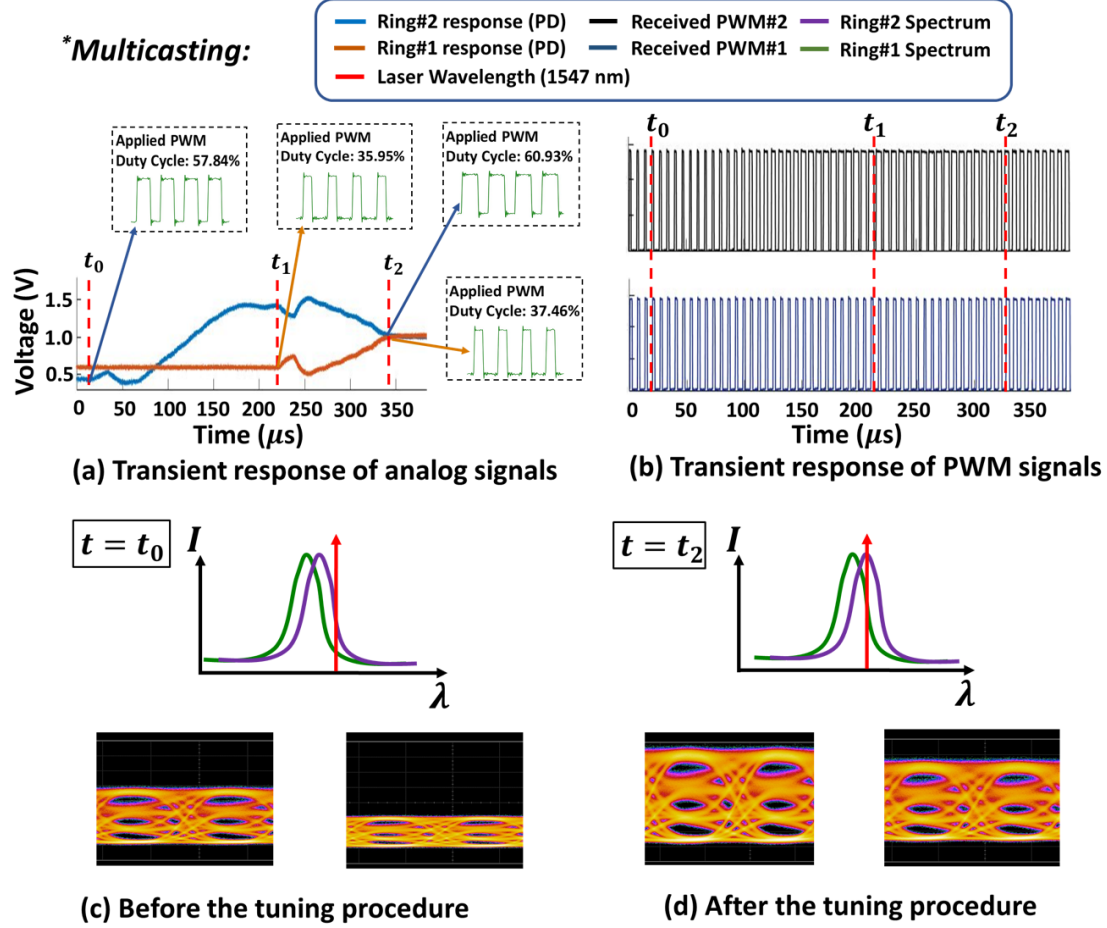


Figure 2.11: (a) Optical transient before and after the feedback tuning procedure for multicasting. (b) Received PWM transient. (c) Corresponding spectrum and eye diagrams before the tuning procedure. (d) Corresponding spectrum and eye diagrams after tuning.

with SERs of $2e-9$ and $2e-7$ (Fig.2.11(c)). After tuning, SERs are improved to $2e-12$ and $3e-11$ for MRR #1 and MRR #2. The eye diagrams and resonances corresponding to input wavelength are shown in Fig. 2.11(d).

Our results show that the single-wire ADC/DAC control and feedback architecture is capable of achieving wavelength switching for unicast and multicast with 20Gb/s PAM4 optical signals in microseconds. The demonstration validates that our novel control and feedback strategy fits the subsystems to obtain high scalability and reliability.

2.5 Chapter Summary

In this chapter, we introduced the control mechanism for MZI, MRR, and MEM-actuated coupler switching cells. We demonstrated optical unicast, multicast, and multiwavelength-select functionalities in cascaded-MRR structures. Both FPGA-based open-loop and closed-loop control are presented. In a stable environment, the open-loop control is sufficient with a calibration of the chip. When ambient temperature varies, a closed-loop control should be applied. The control scheme is agnostic to different types of SiP switches. The MZI and MEMS based spatial switching, and MRR based unicast and multiwavelength-select functionalities are essential for the architectural designs in later chapters.

Chapter 3: Optically Connected Memory for Deep Learning

3.1 Introduction

Recent studies [6, 7] indicate that the deep learning datasets and models are continuously scaling, which will inevitably exceed the memory capacity in today's systems and limit the performance of deep learning applications. As indicated in Chapter 1, there are three major memory challenges regarding deep learning. Here we state them again. First, different deep learning models show varying memory requirements. Second, the memory required during training depends on batch size and optimization strategies. Third, embeddings used in both recommendation systems and language models are huge. Having fixed and preconfigured amount of memory for the maximum memory capacity requirement is inefficient and will become more so. A scalable solution that can adapt to run-time memory requirements is needed to address the memory challenges for future deep learning applications.

Several approaches to tackle the memory capacity issue for large DNNs have been explored. Virtualizing the memory usage of DNNs such that both host and device memory can be utilized by a careful study on the data dependency and network topology of the DNNs is proposed in Ref. [83]. Parallelizing deep learning models across multiple GPUs can be another approach: data parallelism and model parallelism algorithms presented in Ref. [84] show how to distribute large networks among GPUs to relieve the memory capacity limitation. To reduce the communication overhead and achieve better resource utilization, in Ref. [85] a memory-centric architecture is demonstrated in simulation and proposed for future high-performance computing systems. Memory modules are aggregated locally and connected with device nodes using NVLink. Reference [86] proposed using non-volatile memory (NVM) for storing embeddings in deep learning models with caching data in volatile memory to relieve the constraints. The first three approaches tackling the memory

capacity issue with preconfigured and fixed memory resources do not provide a scalable solution to the on-demand memory requirement while the last approach can still be limited by the NVM bandwidth.

Photonic interconnects can enable disaggregated high-bandwidth networks reconfiguring compute and memory resources to meet application requirements in a more efficient and scalable network [87] than those using fixed resource configurations. Memory resources can be pooled and connected to other resources using reconfigurable optical switch fabrics [88]. The system can then be adaptively configured, according to dynamic resource requirements of deep learning applications, to achieve high resource utilization and deliver required system performance. Optically connected memory technique has been demonstrated using custom network interface card [89] with the inevitable overheads in memory-to-network conversions [85]. An optically connected system with emulated processors and a custom memory controller has been reported in [90, 91] without an end-to-end program-level demonstration.

We investigate the feasibility of integrating photonic switched optically connected memory into processing systems to address memory challenges in deep learning. The proposed system architecture enables on-demand allocation of additional memory to processing systems with a constant reconfiguration time that is independent of the required memory size. A “lite” (de)serialization scheme, which avoids heavy memory-to-network conversions and directly (de)serializes memory requests, responses, and data transfers, is proposed to eliminate the network communication overheads. The (de)serialization scheme is compatible with standard memory interface protocol and is applied to memory transfers between the processing system and remote memory nodes via optical links at the program-level. We built a testbed with a processing system node and two remote memory nodes to evaluate the system performance with memory read/write operations. This testbed experimentally demonstrates an end-to-end reconfiguration latency of 2.78 ms and showed a step towards deploying photonic interconnects and optically connected memory for deep learning. Compared to the latency introduced by using storage devices for the DNNs, the proposed system achieves a significant speedup with remote memories.

3.2 System Architecture

Figure 3.1A left depicts the traditional system architecture. Each processing system is composed of CPU, memory, storage, accelerator, and network resources. In order to achieve better accuracy, larger datasets and more complex larger models are being used [6]. Adding more fixed memory modules to the processing system or to the accelerator for large DNNs is not an indefinitely scalable solution that will meet the scaling requirements. Furthermore, incorporating new more advanced hardware with fixed resources cannot guarantee an efficient utilization of compute and memory resources, as the memory capacity requirement for DNN models can vary significantly with applications [92, 93, 23, 26, 94]. We note, therefore, the traditional system architecture for deep learning applications is facing scaling and resource utilization challenges. In our proposed system architecture, as shown in Fig. 3.1A right, the reconfigurable photonic interconnects enable decoupling of additional memory modules from the processing systems and therefore enable flexible allocation of the additional memory capacity to systems or accelerators as required or on-demand. This system architecture breaks through the memory capacity limitation, improves the resource utilization, and is compatible with existing processing systems using designated (de)serialization and memory mapping schemes. Figure 3.1B shows more details of our proposed photonic switched optically connected memory system architecture. The processing system on the left is initially equipped with CPU, memory, accelerator, network, and storage resources. Based upon the memory capacity requirement of the deep learning applications, additional remote memory resources can be connected to the processing system using photonic interconnects through high-speed serial optical links. Helper blocks directly (de)serialize memory requests avoiding potential overheads introduced by network protocols and the NVMs.

Disaggregated memory blocks can be assigned to the processing system using reconfigurable photonic interconnects for two cases. In the first case a processing system occupies the required memory blocks until it finishes the usage of the additional memory capacity. In this case, additional remote memory blocks can be assigned solely to that processing system. The second case occurs

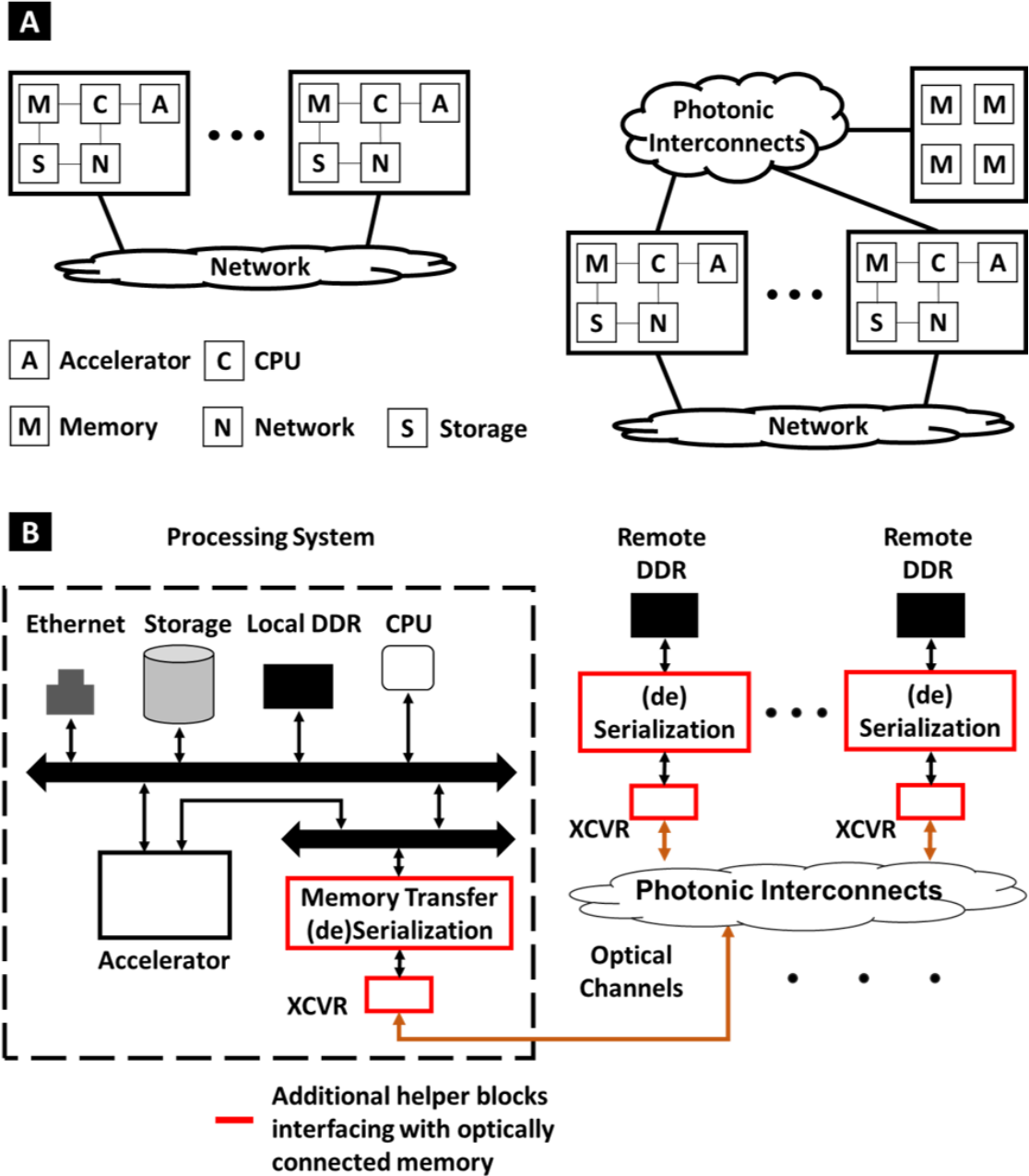


Figure 3.1: (A) On the left, the traditional system architecture with each processing system composed of preconfigured and fixed CPU, memory, storage, accelerator and network resources. In our proposed system architecture, on the right, each processing system using optical I/Os is also connected to a remote memory pool through photonic interconnects. (B) Detailed implementation of photonic switched system architecture with optically connected memory. The processing system includes additional (de)serialization and transceiver (XCVR) helper blocks for (de)serializing memory mapped transactions being transmitted through optical links. On the right, remote double data rate synchronous dynamic random-access memory (DDR) nodes, are also equipped with the (de)serialization and XCVR helper blocks, and the photonic interconnects physically connect remote memory nodes to the processing system.

when multiple processing systems share remote memory nodes. This case depends on the fast switching capability of the photonic interconnects. Remote memory nodes can thus be dynamically selected while applications are running. The optical switching also enables the processing system to access remote memory nodes with limited optical transceiver ports. Examples of these two cases can be found in the following subsection B. In addition, the proposed system architecture can be integrated to current systems with minor modifications to current operating systems.

In this work, we use Xilinx multiprocessor system-on-chip (MPSoC) devices to demonstrate the feasibility of integrating photonic switched optically connected memory into the processing system. Detailed system implementations: (A) a “lite” (de)serialization of memory transfers; (B) mapping remote DDR into the system address space; (C) Silicon Photonic (SiP) switch and control; and (D) accelerator design are presented in the subsections below.

3.2.1 (de)Serialization of Memory Transfers

The MPSoC system uses the AMBA AXI protocol [95] to perform memory read/write operations. To access a locally memory mapped slave device, master devices such as CPU and accelerators can simply launch requests through transaction channels, such as read address, read data, write address, write data, and write response, in order to finish the memory transactions. To access an optically connected remote memory slave, however, the AXI memory mapped channel signals have to be combined and serialized before being transmitted to the remote side through high-speed serial links. We leveraged existing IP blocks designed by Xilinx to achieve the “lite” (de)serialization of the remote memory transfers. Without using any network layer protocol, our scheme directly serializes the AXI channel signals and transfers the high-speed serial signals to the remote nodes through optical links. On the receiver side, the high-speed serial signals are deserialized back to the parallel AXI channel signals.

We primarily used two IP blocks, AXI chip2chip [96] and Aurora 64B/66B [97] IP cores in this system design. The AXI chip2chip core converts the AXI memory mapped channel signals into AXI streaming signals or vice versa and interfaces to the Aurora 64B/66B core. The latter core

utilizes a link-layer protocol, including transceiver initialization, multi-lane handling, and link negotiation for the high-speed serial communication between our optically connected nodes. The AXI chip2chip core can be connected to the AXI interconnects that can be consequently accessed by CPU and accelerators. To achieve an error-free operation, specific transceiver control settings are necessary to be properly configured. These settings depend on the link characteristics. Further details are shown in Section IV.

3.2.2 Map to Local System Address Space

The master CPU and accelerators can only see and communicate with the AXI chip2chip IP blocks in the processing system. In fact, the AXI chip2chip core exposes the remote DDR slave to the local system space. Memory address offsets of the AXI chip2chip and the remote DDR are set to be the same. In this way, CPU and accelerators can seamlessly access the remote DDR as a “local” device. For the case where the processing system occupies multiple memory blocks without optical switching during the application, the remote memory blocks are assigned with different memory address offsets (as they are connected to different chip2chip cores). An example of this case is shown in Fig. 3.2A. Remote DDR #1 node is projected by chip2chip #1 and remote DDR #2 is projected by chip2chip #2. Two remote DDR nodes have different address offset values because they are mapped to separate chip2chip cores. However, for the switching case, the memory address offset of all the remote DDRs is set to be the same. This is due to the fact that CPU and accelerators are accessing the remote DDRs through the same AXI chip2chip core. An example of two remote DDR nodes projected by a single chip2chip core is shown in Fig. 3.2B. The mapping configuration for both cases is one of the modifications to the operating systems.

3.2.3 SiP Switch Control

In this work, we use silicon thermo-optic MRR based 1×8 switch fabrics as spectral-and-spatial de-multiplexers for data routing. We use the MRR to select/drop a specific wavelength to connect communicating nodes. We note that our proposed architecture is agnostic to the choice of

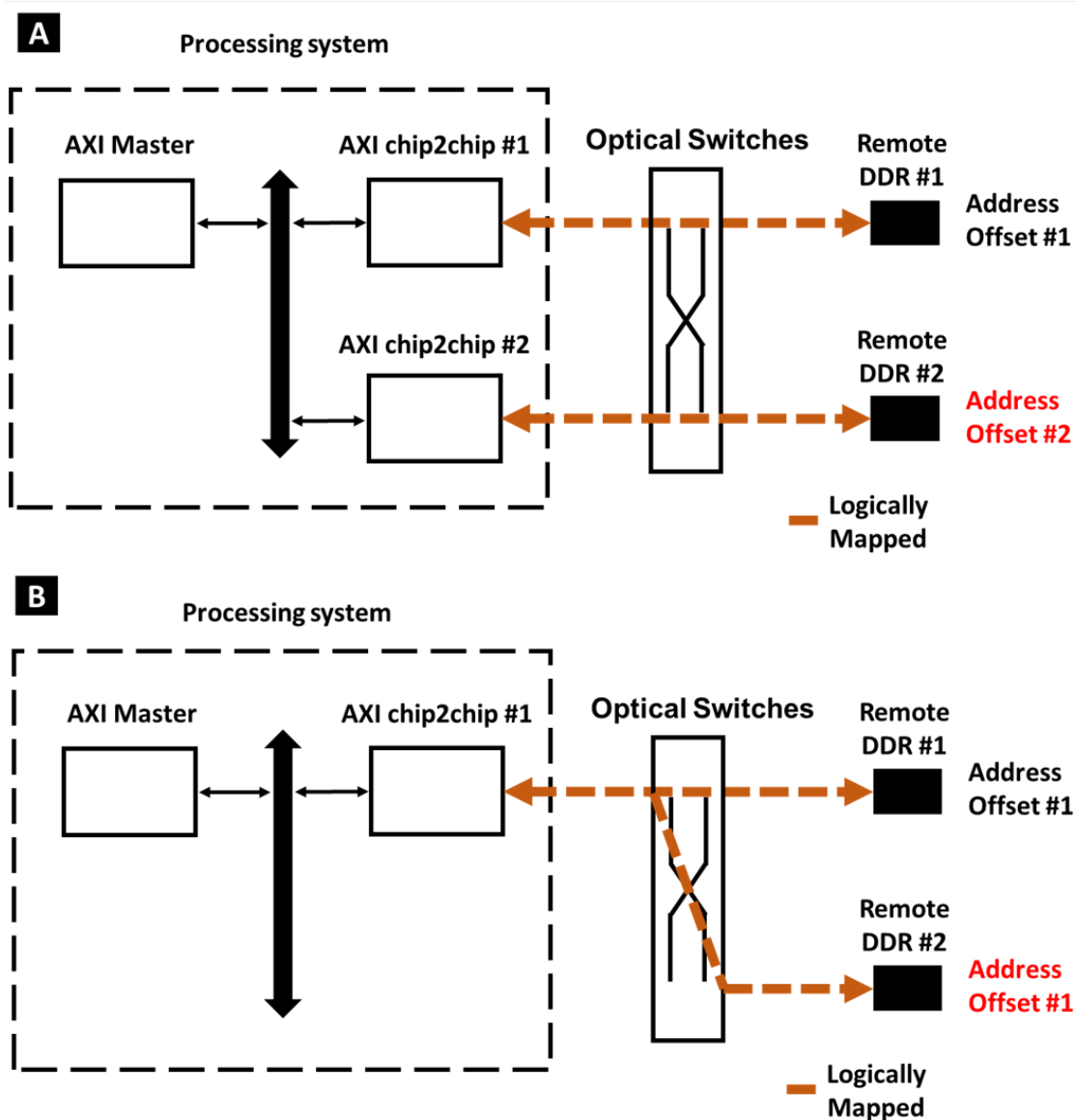


Figure 3.2: (A) An example of case 1, two remote memory resources mapped to two AXI chip2chip cores in the local processing system for the unswitched case after the resources are assigned. Each chip2chip core is assigned with a unique memory address offset (B) An example of case 2, the switching case. Both remote DDR #1 and remote DDR #2 are mapped to the AXI chip2chip #1 in the processing system. They share the same memory address offset.

switching device, although the individual properties of the switch cell choice will have an effect on system performance.

We choose to have an independent switch controller for future system scalability. Controlling high-radix SiP switches generally requires a large number of analog control pins due to the large number of switching elements that forms the switching matrix. A scalable solution is to have a separate switch controller with the required number of analog pins. The processing system will only be required to send configuration requests to the switch controller and the switch controller applies required analog control signals to the switching elements in the SiP switches. We apply this methodology to our proposed system architecture and use group peripheral I/O (GPIO) pins as the interface to the switch controller. These control pins contain 1 bit for triggering and a power of 2 bits for the configurations. Based on the physical configuration required by users or deep learning applications, the processing system will first stabilize the configuration bits and toggle the trigger bit from logic high to logic low to initiate the reconfiguration process. For the switch controller, the procedure is as following: (1) The control logic in the switch controller samples the triggering signal and the configuration bits; (2) if triggered, it reads registers that contain pre-stored digital voltage values associated with each switching element for required configurations and (3) applies the parallel digital voltage values to digital-to-analog convertors (DACs) that bias the switching elements of the SiP switches.

3.2.4 Accelerator Design

We designed a “vanilla” accelerator on the FPGA of the ZCU106 board to further evaluate the feasibility of our photonic switched optically connected memory system architecture. The accelerator uses the standard AXI memory interface and it has the access to remote memory nodes through the AXI chip2chip core. The accelerator functions as a data mover that can “copy” and “paste” data from local DDR to remote DDR or vice versa. Although it does not heavily process the fetched data from either local or remote memories, the functionality of accessing remote memory through a standard memory interface is achieved. The ARM CPU in the processing system initially

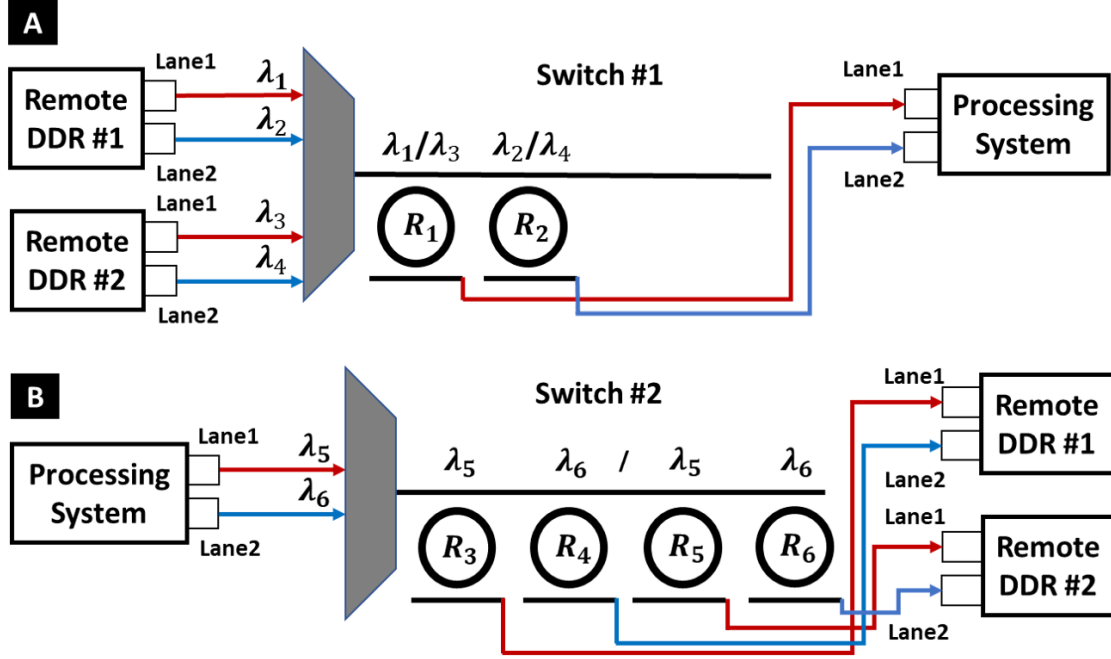


Figure 3.3: SiP switches' configurations for the dynamic access to remote DDRs. (A) Remote memory resources to the processing system direction. (B) The processing system to remote memory resources direction.

comes with AXI interface and it does not require additional implementations.

3.3 Testbed

We built an experimental testbed to evaluate the optical links and switching characteristics, and to demonstrate the feasibility of integrating SiP switches and remote DDRs into the processing system for DNNs. It includes one processing system node dynamically connecting two remote DDR memory blocks.

Two SiP switches connect the processing system to the remote DDR nodes. In this specific implementation of our architecture only a 1×2 switch and a 1×4 switch are required, although we used 1 *times* 8 SiP switches for the experiment. As we are only accessing the first MRRs, the experimental results are not impacted. Based on our system configurations, two MRRs in one of the 1×8 SiP switches are used for the direction from remote DDR nodes to the processing system and four MRRs in the other 1×8 SiP switch are used for the processing system to remote DDRs

direction. We label them as 1×2 and 1×4 switches in the rest of the chapter. In addition, each optical link contains two bundled lanes.

Figure 3.3A shows the direction from remote DDR nodes to the processing system. If the processing system requires the connection to remote DDR node #1 then MRR #1 and MRR #2 in the 1×2 switch are tuned to select and forward λ_1 and λ_2 to the processing node. For the connection to the remote DDR #2, λ_3 and λ_4 are selected. Figure 3.3B shows the other direction for data transactions. If the system is configured as remote DDR #1 node being connected to the processing system node, the first two MRRs connected to the remote DDR #1 node in this 1×4 switch will drop λ_5 and λ_6 . When the remote DDR #2 node is acquired by the processing system, MRR #3 and MRR #4 in the 1×4 switch are detuned from λ_5 and λ_6 to allow the light to pass through while MRR #5 and MRR #6 are tuned to drop and forward the light to the corresponding receiver ports of the remote DDR #2 node.

Figure 3.4A shows the experimental setup. Two Xilinx ZCU106 and a Terasic TR4 evaluation boards are used to evaluate the system. One of the ZCU106 boards contains both the processing system and remote DDR #1 nodes. The physical connection is only through the optical link that can be steered by the SiP switches. The other ZCU106 only comprises the remote DDR #2 logics. Each remote DDR node contains a 2 GB 64-bit wide DDR4 memory system. Six transceivers in total are used to support multi-lane optical communications. Each link contains two lanes and each lane operates at 10 Gb/s data rate. The maximum throughput for the serial link between the processing system and a remote DDR node can reach up to 20 Gb/s. Four C-band SPF+ transceivers, with wavelengths at 1545.32 nm (λ_1), 1546.92 nm (λ_2), 1553.33 nm (λ_3) and 1554.94 nm (λ_4), are used for the two remote DDR nodes to transmit data to the processing system, and two wavelengths at 1554.94 nm (λ_5) and 1556.56 nm (λ_6) are used for the opposite direction. Optical signals are combined by the multiplexers (MUX) and then enter the SiP MRR based switch chips. The polarization controllers (PC) change the polarization of the light of each lane to maximize the optical power being coupled in to and out of the SiP chips. An erbium doped fiber amplifier (EDFA) is necessary to compensate the loss due to the grating couplers of the SiP switch chips.

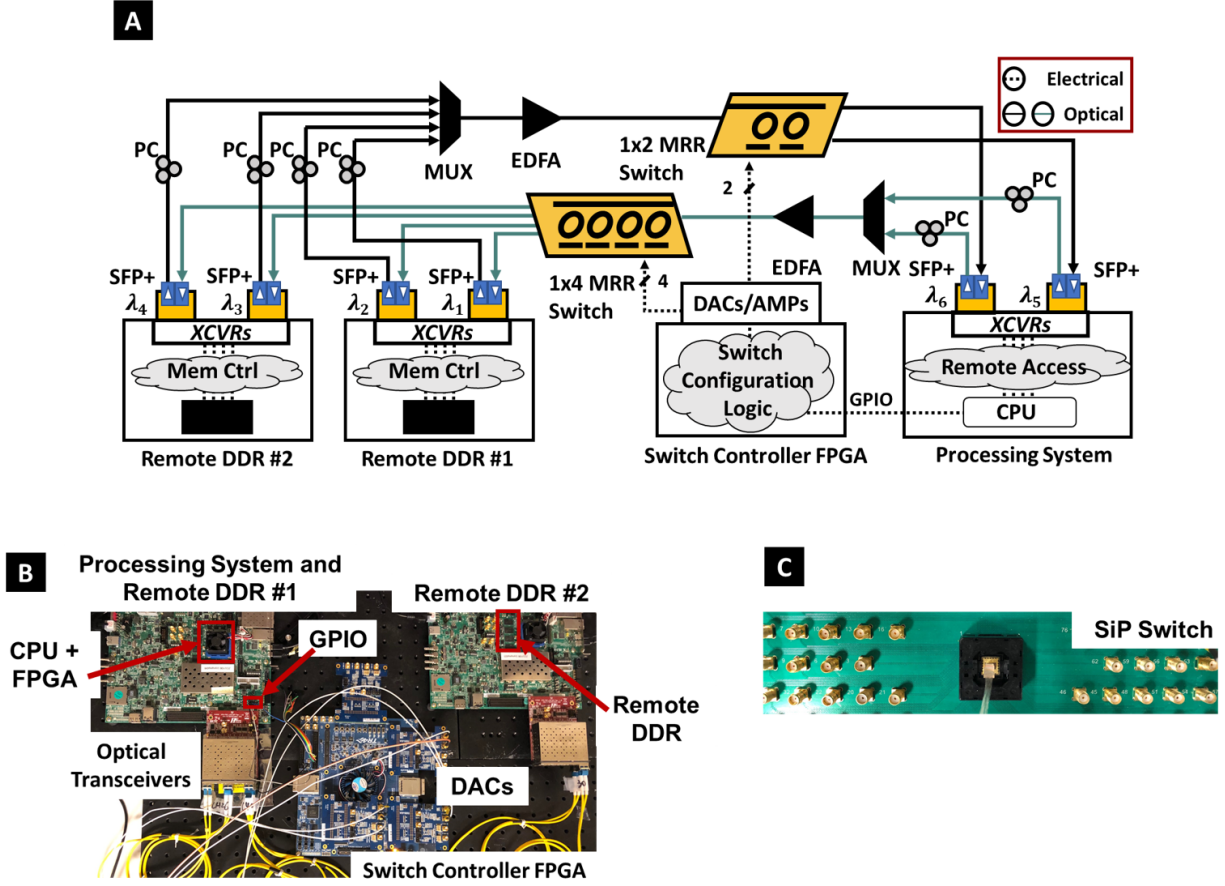


Figure 3.4: (A) Experimental setup demonstrating a case of photonic switched optically connected memory system with dynamic allocation of remote DDR resources to the processing system. (B) Key hardware components. One Xilinx ZCU106 board containing the processing system and the remote DDR #1 nodes, another ZCU106 board containing only the remote DDR #2 node, and the TR4 switch controller FPGA board. (C) A packaged SiP MRR based switch with electrical SMA interface.

The processing system sends configuration requests to the switch controller FPGA, on the Terasic TR4 board, which configures each MRR by tuning the resonance of each MRR with bias voltage through DACs and electrical amplifiers (AMPs). The electrical amplifiers are used to provide sufficient voltage levels to the MRRs. The configuration and trigger signals are transmitted through GPIO pins from the processing system ZCU106 board to the TR4 board. Figure 3.4B illustrates the key hardware components that enable the evaluation of the system. CPU, FPGA, remote DDRs, GPIO, optical transceivers, switch controller and DACs are used for the evaluation of optical link

and switching characteristics. A packaged SiP chip on a printed circuit board with SMA interface is shown in Fig. 3.4C.

3.4 Experiment and Results

3.4.1 Optical Spectra

We first demonstrate that the SiP MRR based switches are capable of supporting the multi-lane optical communications required for the two different physical memory access topologies. Figure 5A shows the optical spectra at the drop port of each MRR configured for prioritizing the physical connection between the processing system to the remote DDR #1 node. MRR #1 and MRR #2 are tuned to drop the optical wavelengths at 1545.32 nm (λ_1) and 1546.92 nm (λ_2) for the lane #1 and lane #2 from the remote DDR #1 node. The received optical power of the data signal at the corresponding receiver ports is -15.60 dBm and -18.32 dBm respectively. MRR #3 and MRR #4 are configured to select and forward the wavelengths at 1554.94 nm (λ_5) and 1556.56 nm (λ_6) from the processing system to the remote DDR #1 node. The received optical power for lane #1 and lane #2 are -17.48 dBm and -16.39 dBm respectively. For the plots of MRR #1 to MRR #4, the highest peak is the optical data signal and other peaks are crosstalk from adjacent optical channels. For the plots of MRR #5 and MRR #6, the peaks show leakage power from previous MRRs, MRR #3 and MRR #4.

Figure 5B shows the optical spectra at the drop output of each MRR for the second case where the remote DDR #2 node is connected to the processing system. The optical power received by the processing system at 1553.33 nm (λ_3) and 1554.94 nm (λ_4) is -14.96 dBm and -18.25 dBm respectively. MRR #3 and MRR #4 are detuned to allow the light to pass through these MRRs and the light can be dropped by MRR #5 and MRR #6. The received optical power at the receivers of DDR #2 node are -20.3 dBm and -18.3 dBm respectively. We ensured the received optical signal power is above the receiver sensitivity of -23 dBm.

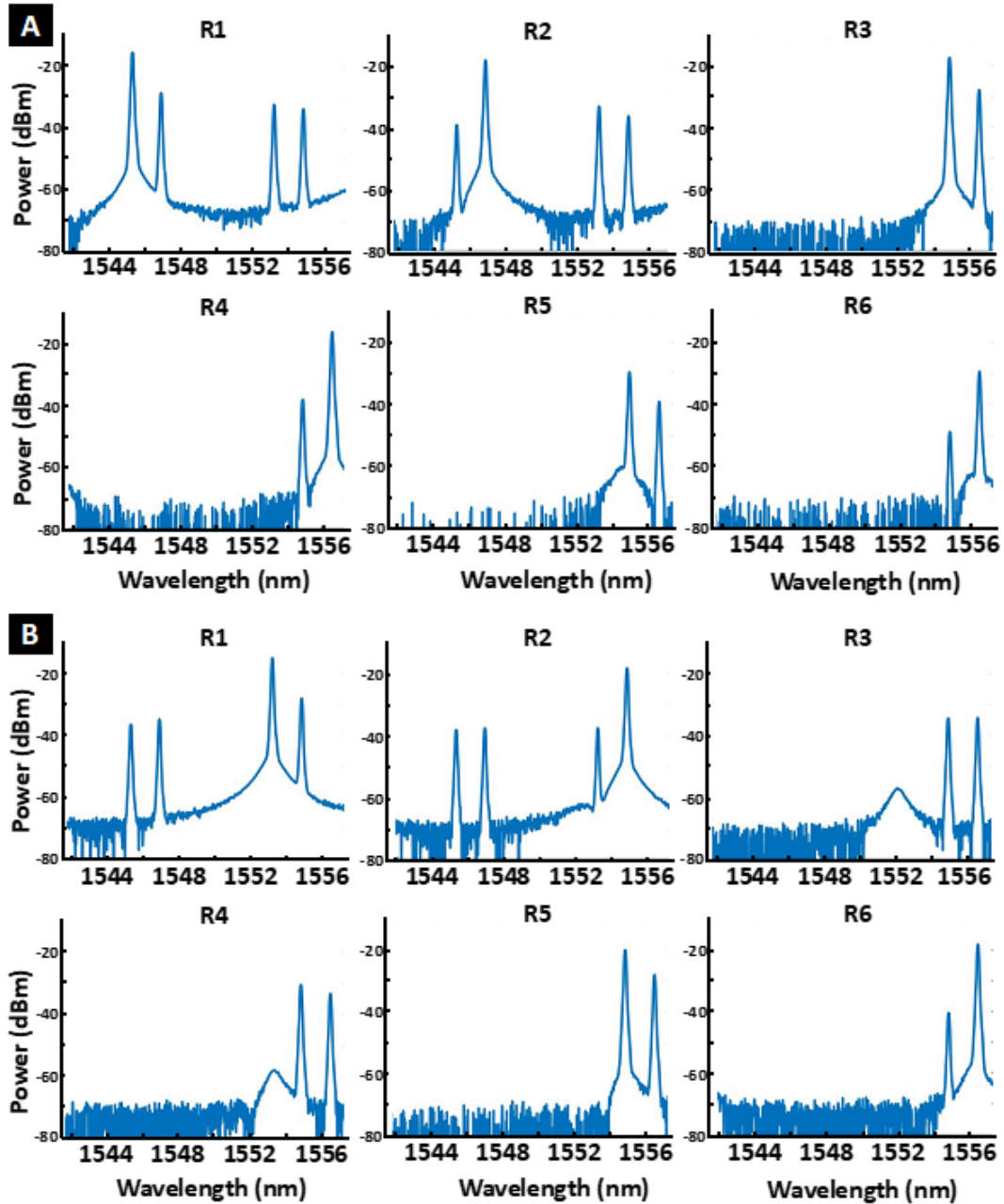


Figure 3.5: Optical spectra at the drop port of each MRR for two different configurations. (A) Two SiP switches configured as the processing system connecting to the remote DDR #1 node. (B) Two SiP switches configured as the processing system connecting to the remote DDR #2 node. (In this figure, the MRR numbers are consistent with the MRR numbers shown in Fig. 3.3.)

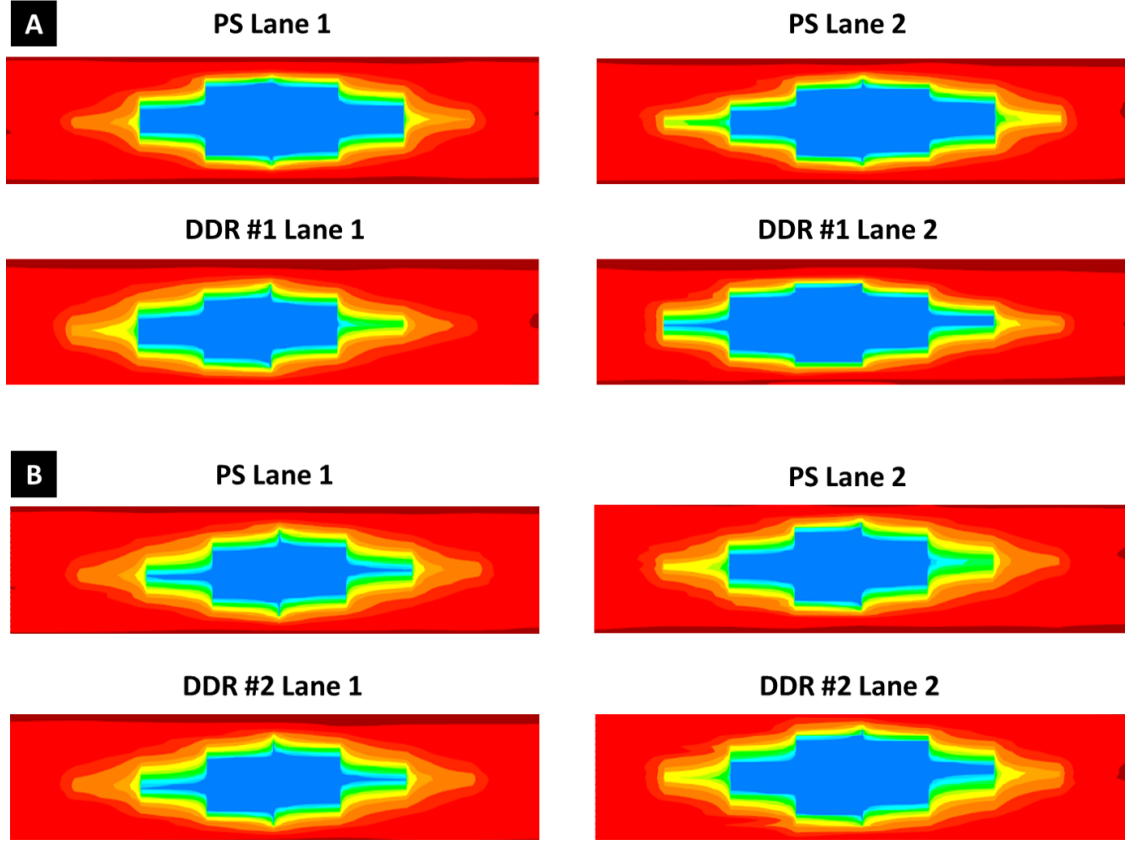


Figure 3.6: Screen shots of open eye diagrams of connected receiver ports at 10 Gb/s PRBS-31. (A) Two SiP switches configured as the processing system (PS) connecting to the remote DDR #1 node. (B) Two SiP switches configured as the processing system connecting to the remote DDR #2 node.

3.4.2 Eye Diagrams

Data transmission at 10 Gb/s non-return-to-zero (NRZ) on-off keying (OOK) using $2^{31}-1$ pseudo-random bit sequence (PRBS-31) was performed to extract transceiver settings for the Aurora 64B/66B IP core. With transmitter driver swing at an amplitude of 647 mV_{PPD}, pre-cursor TX pre-emphasis of 0.68 dB and post-cursor TX pre-emphasis of 1.16 dB, error-free operations over the optical links are achieved. All the connected paths for the two different configurations show clear eye-openings as shown in Fig. 3.6.

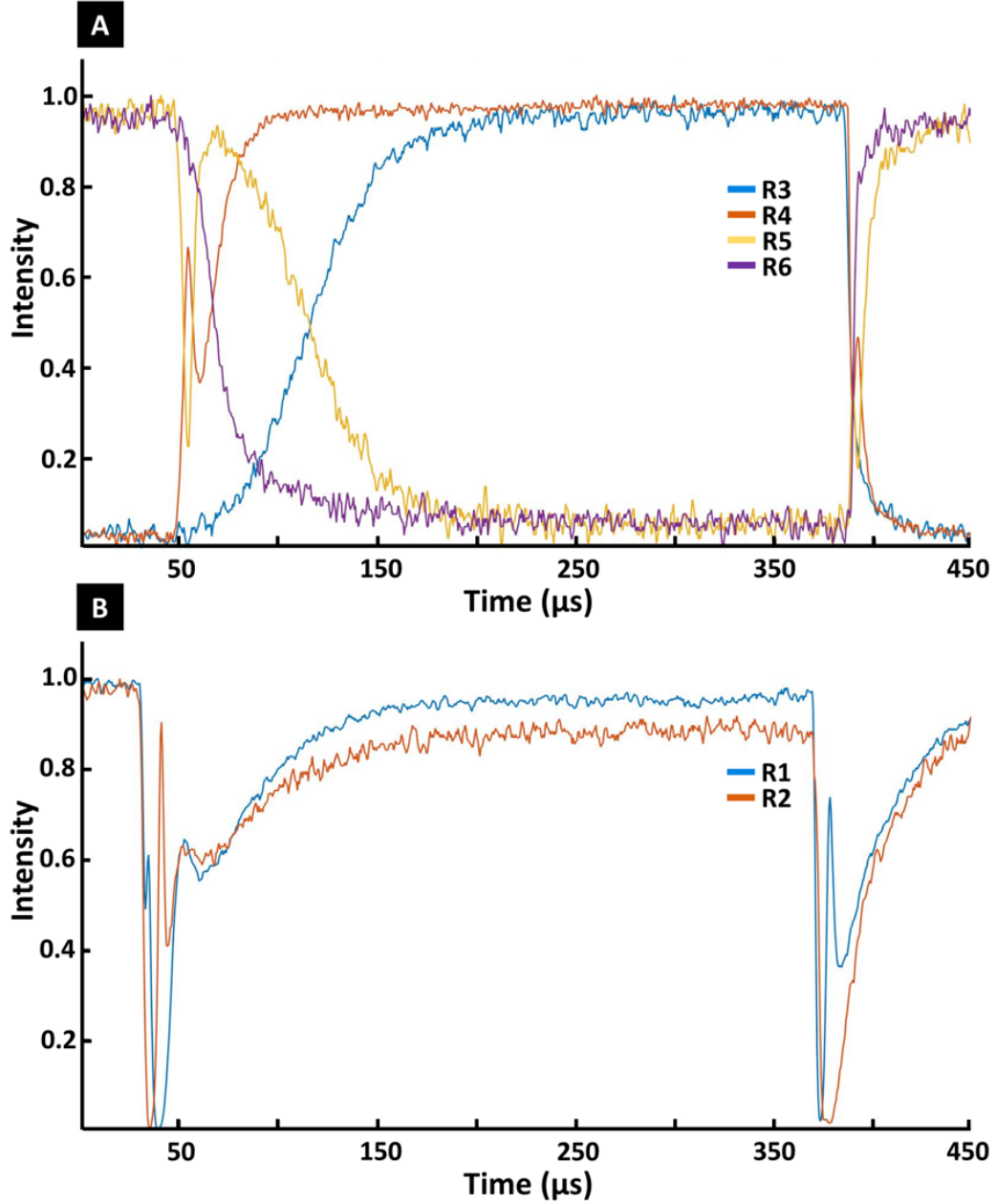


Figure 3.7: Screen shots of open eye diagrams of connected receiver ports at 10 Gb/s PRBS-31. (A) Two SiP switches configured as the processing system (PS) connecting to the remote DDR #1 node. (B) Two SiP switches configured as the processing system connecting to the remote DDR #2 node.

3.4.3 Switching Time

We performed measurements of two switching cases between two configurations: (1) the remote DDR #2 node connected to the processing system, and (2) the remote DDR #1 node con-

nected to the processing system. The first switching case is changing from configuration #1 to configuration #2 and the second switching operation happens $330\ \mu\text{s}$ after the first switching operation, which is changing from configuration #2 to configuration #1. In Fig. 7, we show the transient responses of the received optical power, normalized individually for each MRR.

As shown in Fig. 3.7A, the first switching case starts at the time that approximately equals to $50\ \mu\text{s}$. We notice that MRR #4 experiences faster rise time than MRR #3 and becomes stabilized within a shorter time. The local maxima and the local minima of the orange curve are due to the fact that MRR #4 passes through $1554.94\ \text{nm}$ (λ_5) during the first switching process. MRR #5 and MRR #6 are initially tuned at $1554.94\ \text{nm}$ (λ_5) and $1556.56\ \text{nm}$ (λ_6), and the control bias voltages are not changed during the process, thus the transient response of MRR #5 is reciprocal to the superposition of the transient responses of MRR #3 and MRR #4. The transient response of MRR #6 is reciprocal to MRR #4 only. For the configuration #2 to configuration #1 case, MRR #3 and MRR #4 are detuned to allow the optical signals to pass through, and the transient responses can be observed at the time approximately equal to $380\ \mu\text{s}$. The limiting factor of the switching operation is the slowest transient response of all the responses. We can see from Fig. 7A that the rise time of MRR #3 for the switching from configuration #1 to configuration #2 is the slowest transient response and the latency is approximately $119\ \mu\text{s}$. We have, however, shown that the thermo-optic switching time can be as low as $1.2\ \mu\text{s}$ with optimized driving circuitry [54].

Figure 3.7B illustrates the transient responses at the receiver ports for MRR #1 and MRR #2 in the two switching scenarios. Since both MRR #1 and MRR #2 are dropping optical signals for two configuration cases, the transient response of each individual MRR is expected to fall first and then rise back during the switching process. As we find from Fig. 7B, the slowest transient response is approximately $107.5\ \mu\text{s}$ at the receiver port for MRR #1 in the first switching case.

3.4.4 End-to-end Reconfiguration Time

The system end-to-end reconfiguration latency consists of (1) the time for AXI chip2chip and Aurora 64/66B cores to reset, (2) optical switching time, and (3) link re-negotiation time. To

reconfigure the physical connections between the processing system and remote DDR nodes, the AXI chip2chip and Aurora 64/66B cores in the processing system are required to be put into reset state. This reset action will also be propagated to the remote DDR end to restart the link-renegotiation process. The reset process and the link-renegotiation process are described in [96, 97]. One requirement for this process is that the asserted reset state needs to last at least 128 user clock cycles and we chose to set the cores to be in reset state for 2 ms. The optical switching time shown in the previous section is approximately $119\ \mu\text{s}$ and we chose to wait $330\ \mu\text{s}$ to ensure the optical link is stabilized. The reset was then released and the link-renegotiation process started. This renegotiation time was measured to be 0.45 ms. In total, the end-to-end reconfiguration time was 2.78 ms.

3.4.5 Application and Execution Time

We built a Linux kernel image based upon the system implementation using Xilinx PetaLinux tool and booted the operating system with Ubuntu 18.04 filesystem on the Xilinx ZCU106 board. The kernel image is stored in the SD card boot partition while the filesystem is stored in the hard drive root partition. The hard drive is connected to the processing system through SATA interface.

We evaluated the system performance by measuring the latencies of loading data from storage to local memory, storing data from local memory to storage, loading data from remote memory to local memory, storing from local memory to remote memory, and classifying an image on the ARM Cortex CPU. A VGG16 model was pretrained using TensorFlow in Python and its parameters, such as weights and biases for each layer in the network, are also saved in the hard drive. A feedforward implementation of the neural network including the convolutional and fully-connected layers is coded in the C programming language, thus the processing system is capable of running a C program to load the parameters and classify an image using the pretrained VGG16 model on the ARM CPU. The VGG16 model contains 13 convolutional layers and 3 fully-connected layers with 138,357,544 parameters and we use 32-bit floating point data type for each parameter. Thus, the total size of the VGG16 is approximately 528 MB. The loading time from the hard drive to

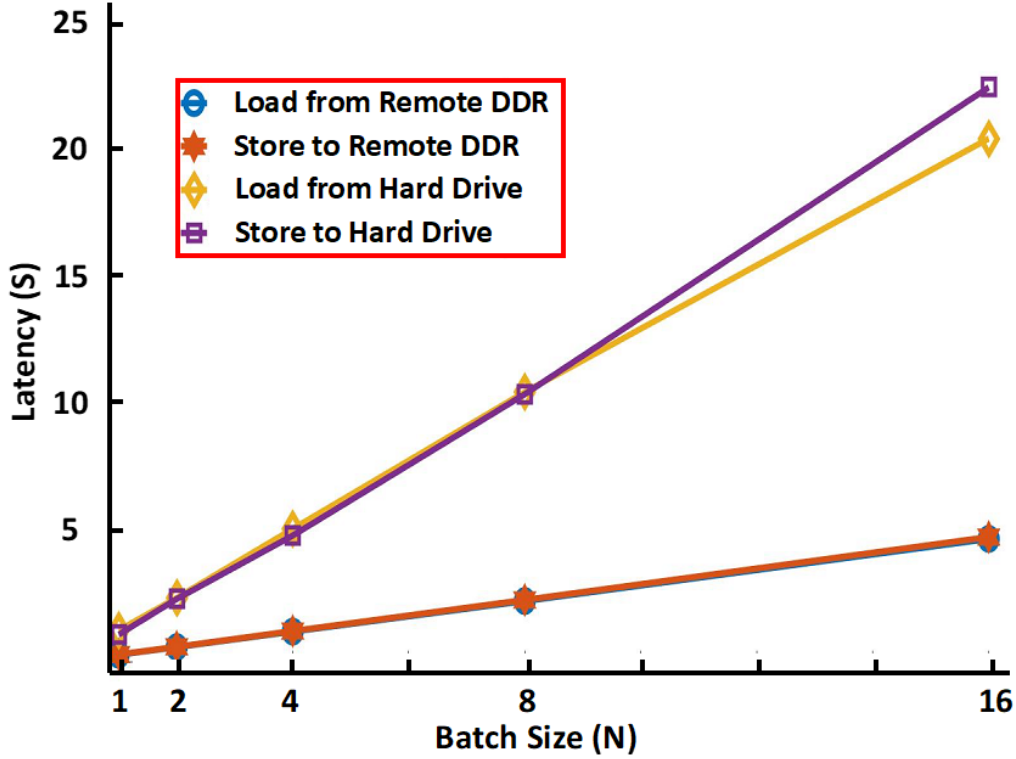


Figure 3.8: Loading/storing latencies using hard drive and remote DDR memory of different batch sizes.

the local main memory is 5.70 s for the entire VGG16. The execution time to classify an image is 63.34 s on the ARM CPU.

To measure the latencies of using remote DDR for storing/loading parameters, i.e. weights and biases, we use our designed accelerator in a standalone design (without the operating system). The time for storing 528 MB data, the same size as the VGG16, from the local contiguous memory allocation (CMA) region to the remote DDR takes 1.40 s for the accelerator and loading the data from remote DDR to the local CMA region takes 1.34 s.

The accelerator’s equivalent throughput for loading from the remote memory to local memory is 3.31 Gb/s, and 3.16 Gb/s for storing. The limited throughput is due to the fact that the designed accelerator operates at 250 MHz with 32-bit AXI data channel width, which can theoretically achieve up to 7.8 Gb/s without overhead. In addition, the accelerator performs “copy” and “paste” operations which lead to an overhead factor of approximately 0.5 over the entire system. By

increasing the clock frequency and data channel width of the accelerator, higher throughput can be achieved.

During training, memory space is required to store each layer’s output and its corresponding gradients. The space required for gradients is the same size as the layer’s output for backpropagation when stochastic gradient descent (SGD) [24] optimization strategy and ReLu [98] activation function are used. To evaluate the feasibility of our proposed architecture for increasing the memory capacity for training, we performed a forward propagation of the VGG16 with a batch size of 1, 2, 4, 8, and 16 images in the software. Based upon the memory requirement for training, we stored/loaded the intermediate layer results and randomly initialized gradients to/from both remote memory and the hard drive for the purpose. The intermediate results include the output of each convolutional layer, max pooling layer and fully-connected layer. There are 15,087,080 elements of intermediate layer results per image to be stored for backpropagation. Considering the gradients, there are in total 30,174,160 elements per image that are being stored/loaded during the process. The time for storing to the hard drive is 1.14 s, 2.49 s, 4.98 s, 10.53 s, and 22.58 s, respectively. For loading from the hard drive, the latencies are 1.26 s, 2.53 s, 5.24 s, 10.61 s, and 20.57 s, respectively. As expected, the latencies for storing/loading using remote DDR are less than using the hard drive in the testbed. The storing latencies using the remote memory are 0.31 s, 0.61 s, 1.23 s, 2.45 s, and 4.92 s, while the loading latencies are 0.30 s, 0.60 s, 1.21 s, 2.41 s, and 4.85 s, respectively. Figure 3.8 compares the latencies of using hard drive and remote DDR memory and shows that the required memory space for layer output and layer gradients grows with the batch size. We note that larger batch size will require more memory and the memory requirement is also related to the use of other optimizers [16], but the functionality of our architecture and the remote memory remains the same. Table 3.1 lists the results for the system performance measurements.

Figure 3.9 compares the three scenarios for the test case of inference: processing system loading from storage, processing system with remote DDR and optical interconnect, accelerator with remote DDR and optical interconnect. The total execution time consists of both compute time and the time for data access. For the latter, we can achieve a speedup of 4.3 when loading the data

Table 3.1: System Performance Measurements

Operations	Latency
Optical switching	119 μ s
End-to-end reconfiguration	2.78 ms
Load VGG16 (528 MB) from hard drive to local DDR memory	5.70 s
Load 528 MB data from remote DDR to local DDR CMA region (accelerator)	1.34 s
Store 528 MB data from local DDR CMA region to remote DDR (accelerator)	1.40 s
Load intermediate results and gradients from hard drive to local DDR memory (batch size of 16)	20.57 s
Store intermediate results and gradients from local DDR memory to hard drive (batch size of 16)	22.58 s
Load intermediate results and gradients from remote DDR to local DDR CMA region (batch size of 16)	4.85 s
Store intermediate results and gradients from local DDR CMA region to remote DDR (batch size of 16)	4.92 s
Classify an image using VGG16 on ARM CPU	63.34 s

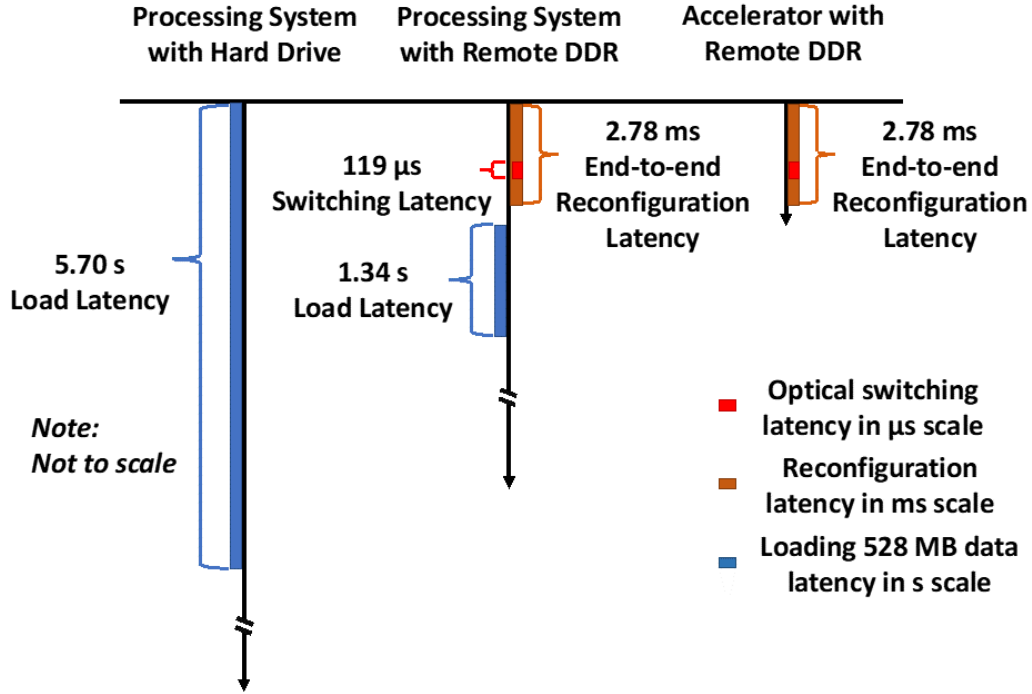


Figure 3.9: Timelines comparing system latencies in different scenarios for switching case #2.

from remote DDR compared to loading from the storage device to the local DDR. The end-to-end reconfiguration latency we observed is much shorter than the loading time therefore we use 1.34 s as the total time for the processing system to load the data from the remote memory. In the case of the accelerator, the end-to-end reconfiguration time is the only one considered as the accelerator can directly access the remote memory without loading. We note that the optical reconfiguration time is a constant overhead independent of the data size. With increased data size the impact of the overhead is amortized.

3.5 Discussion

Optical switching technology enables reconfigurable disaggregation allowing the processing system to dynamically access additional memory resources. In order to successfully integrate the photonic switched optically connected memory into the system, several requirements for the optical switches need to be taken into consideration including: optical power budget, reconfiguration time, power consumption and scalability.

The optical power budget available is based on the receiver sensitivity and the optical power launched by the transmitter. The insertion loss of the optical switches should be well below this if the system has no optical amplification. If the insertion loss of the switch and additional losses in the link go beyond this optical power budget, optical amplification is required, which is generally not desirable due to energy and cost considerations, although recent work with semiconductor amplifiers has shown promise [99]. The extinction ratio (ER) of the optical switch also depends on the optical transceiver. For a transmitter ER of 3.5 dB, we measured more than 10 dB power suppression ratio of the optical signal power to the optical leakage power which can guarantee error-free operation.

End-to-end reconfiguration latency is an important network parameter. This parameter includes optical switching time, transceiver reset and link negotiation. In order to not introduce excessive overhead, the optical switch reconfiguration should not occupy more than ten percent of the entire reconfiguration latency. In this case we have shown that the optical switch reconfiguration time is not detrimental to the system performance. To decrease the overhead of the optical switching and link reconfiguration latency, advanced high-speed devices could be employed. Electro-optic switches and burst-mode transceivers can be deployed in the system. Electro-optic silicon photonic switches provide nanosecond-scale reconfiguration time [51] and sub-nanosecond clock and data recovery has been demonstrated in an optically switched link via clock phase catching [100]. The achievable end-to-end reconfiguration latency can thus be reduced to the nanosecond scale.

The power consumption of the optical switch should be a small fraction of the power consumption of the entire system. The state-of-the-art GPU [101] can consume up to 280 W while reported silicon photonic switches [102], are in the range of Watts and are therefore relatively power efficient when integrated into the system to support dynamic memory resource allocation. For example a 32×32 MZI-based switch consuming a power of 1.9W [47]. The switch fabric used in this experiment consumes approximately 10 mW per MRR.

Although we demonstrated a 1×2 switching scenario in the testbed, larger $N \times M$ optical switches in application dependent topologies would support the system requirements, depending

on the number of compute/accelerator nodes (N) and the remote memory nodes (M) within the subsystem. Ref. [85] indicates a use case of 8 compute and 8 memory nodes. A full analysis of the relationship between the radix/topology of the optical switch and the overall system performance/cost can be performed using the same methodology as shown in our previous work [42], for specific applications and switch architectures.

Our experimental testbed was designed to experimentally demonstrating the proof-of-concept functionalities of our proposed system architecture. Although we used legacy SATA based storage devices in our testbed, commercially available storage drives can support up to 2,375 MB/s throughput (Amazon Web Service [103]). In order for our proposed architecture to demonstrate comparable speedup using commercial high-end storage devices, one would build an optical system with comparable high-end bandwidth optical I/Os and optimized transceiver circuitry. Multiwavelength terabit optical links are under development with state-of-the-art silicon transceivers capable of modulating [104] and detecting [105] at over 100 GHz bandwidth.

In summary, our proposed photonic switched system architecture demonstrates the concept of using dynamic allocation of memory to tackle the scaling challenge of deep learning. Our test cases demonstrate the capability of increasing memory capacity at the program-level using an architecture based on MRR optical switches, FPGA processing systems, and optically connected DDR memories. The designed “lite” (de)serialization and memory mapping scheme show a path towards lowering the system latency, a critical metric for disaggregated systems. The independent switch controller is scalable and is able to be applied in the systems requiring large number of switching elements as long as they are controlled by biasing voltages. The proposed system architecture shows a significant step toward deploying photonic interconnects and optically connected memory for deep learning applications. More generally, with specific optimizations the approach would also be applied to other workloads that face the same memory challenges.

3.6 Chapter Summary

We demonstrate a proof of concept system architecture, showing the functionality of photonic switched optically connected memory for large DNNs in deep learning. It features dynamic allocation of additional memory to the processing system and a constant reconfiguration latency. The experimental testbed demonstrates real memory transactions between the processing system and remote memory nodes. We measured a $119\ \mu\text{s}$ latency for optical switching and an overall $2.78\ \text{ms}$ latency for the end-to-end reconfiguration. Our results and silicon-based high-bandwidth I/O capabilities show the feasibility of using photonic switched optically connected memory to solve the memory challenges in future deep learning applications.

Chapter 4: Photonic Switched Architectures for Distributed Deep Learning

4.1 Introduction

As discussed in Chapter 1, the demand for better DL models has resulted in a rise of more complex models that support larger dataset sizes to improve these deep neural networks. Today, DL workloads are taking a large proportion of the computation in HPC and datacenters, and will continue to grow. These trends shift the bottleneck from computation resources to networks, and require high performance and reconfigurable interconnects to sustain the continual growth of the distributed deep learning applications.

Flexible photonic switched networks have the capabilities to perform topology reconfiguration and many research works have explored reconfigurable network architectures using OCSs. These OCS-based architectures employ various different technologies, such as 3D MEMS [106, 71], silicon photonic switches [74], wireless transceivers based on free space optics [73, 107], RotorSwitch [108], and tunable lasers [109]. Early architectures of reconfigurable network such as Helios [71] used OCSs to build a hybrid optical/electrical architecture to serve bandwidth-bound large flows using the OCS network while serving the latency-bound small flows with static electrical packet switches (EPSs). Later works such as ProjecToR [73] and RotorNet [108] used customized switching prototypes to build flatter network topologies where the top-of-rack (ToR) switches are directly connected with a single layer of OCSs for higher energy efficiency. Meanwhile, silicon photonic (SiP) switches have also been proposed as another solution that could provide power-efficient high bandwidth scaling at low fabrication cost. Flexfly [107] and Flexspander [110] placed SiP switches in between clusters/groups of EPSs to achieve better scalability.

In addition to applying these reconfigurable network architectures to traditional HPC workloads, various works in the literature have explored employing them under distributed machine

learning settings as well. Truong et al. [111] have proposed using a hybrid electrical/optical architecture, similar to Helios [71], to serve long-lived DL training communications using the OCS network while using the EPS network for smaller messages. Evaluation with real DL workload shows significant communication speedup when employing the hybrid architecture. Lu et al. [112] have proposed to build a hierarchical network, similar to Flexfly [74], for distributed machine learning applications. Results show that X-NEST outperforms RotorNet16 across different DL workloads and performs similarly to fat trees with fewer hardware components.

While many reconfigurable network architectures have been explored in the past, prior work has typically proposed architectures with reconfigurability at a single network layer (e.g. between ToR and aggregation EPSs [113] or between dragonfly groups [74]). We propose our reconfigurable SiP architecture [114] that uses SiP switches between servers and ToR, and between ToR and aggregation EPSs in a fat tree topology. This architecture introduces two unique network functionalities: (1) server-regrouping between servers and ToR switches to recover job-level traffic locality, and (2) bandwidth-steering between ToR and aggregation layers to maximize traffic retention at the lower fat tree layers. We demonstrate an improvement in overall network performance for distributed ring all-reduce and parameter server deep learning training algorithms. An optimized SiP switch control scheme is presented to simplify the control implementation complexity and to achieve better integration of the SiP switches into large-scale systems. In our experimental hardware testbed [114], we present new results demonstrating that regrouping servers and steering network bandwidth can result in more efficient execution of the distributed deep learning workloads. We report a $1.9\times$ to $3.6\times$ performance improvements depending on different distributed training strategies and test cases. In this paper we also present new system-scale simulations. We perform server regrouping and bandwidth steering on a large-scale tapered fat tree with 1024 compute nodes. Our simulation results show that server regrouping can deliver up to $2.3\times$ flow throughput improvement for a $2\times$ tapered fat tree and a further 11% improvement when higher-layer bandwidth steering is applied.

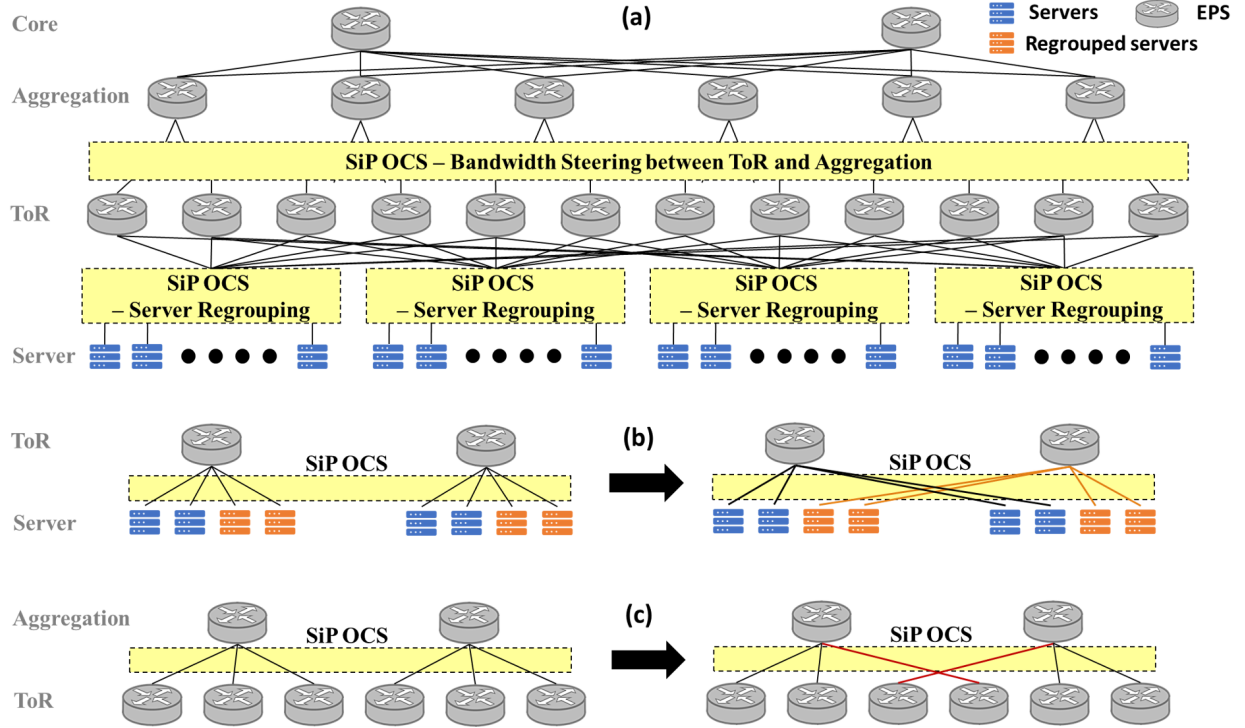


Figure 4.1: (a) System architecture demonstration with server nodes arranged in the fat tree topology to show SiP switch-based server regrouping and higher-layer bandwidth steering. (b) An example of before (left) and after (right) server regrouping. (c) An example of before (left) and after (right) bandwidth steering above the ToR.

4.2 System Architecture and SiP Switch Control

4.2.1 System Architecture

Distributed deep learning training workflows, including data parallelism and model parallelism, show strong communication patterns with high-bandwidth requirement between server nodes. We demonstrate our proposed system architecture on synchronized ring all-reduce [115] and asynchronized parameter server [20] data-parallel techniques. Figure 4.1(a) illustrates our proposed system architecture. It consists of EPSs, SiP-based OCSs, and servers to demonstrate the capabilities of server regrouping and network bandwidth steering. By using SiP OCSs between servers and ToR EPSs, this architecture allows servers with intense communication requirements to be grouped locally within the same ToR-switch, thereby reintroducing traffic locality between physically distant

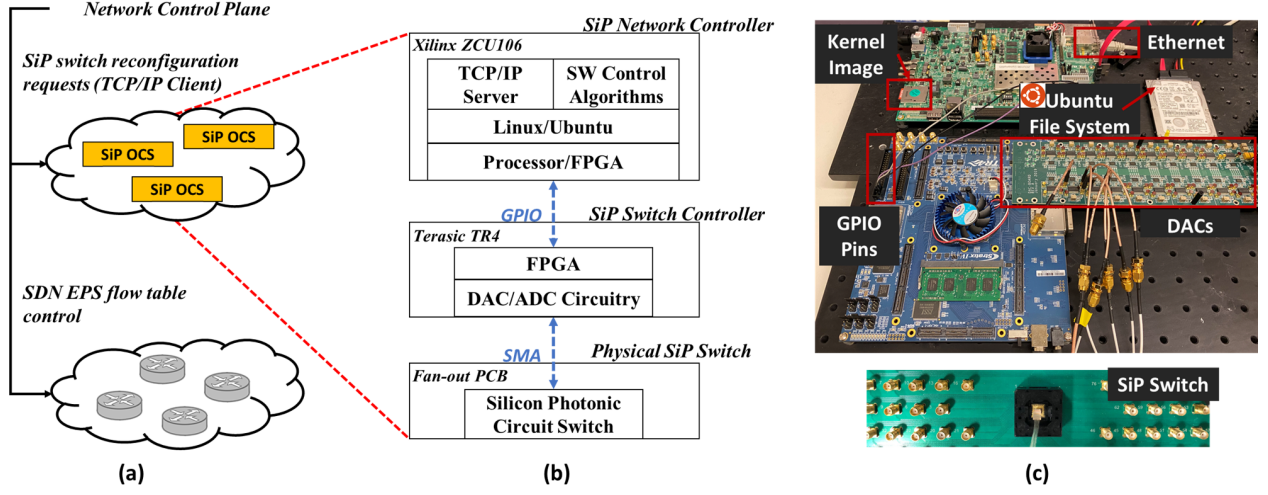


Figure 4.2: (a) Overall network control plane. (b) SiP OCS subsystem including the SiP network controller, SiP switch controller, and SiP switches. (c) The SiP network controller board (ZCU106), SiP switch controller board (TR4), and PCB holding a packaged SiP switch.

servers. An example of regrouped servers is shown in Fig. 4.1(b). With the demand of traffic between servers (in orange), the SiP OCS is capable of dynamically changing the connectivity and connecting the regrouped servers (orange) under the same ToR EPS. Due to the limited port count of the SiP OCS, it is not feasible to realize all-to-all ToR connectivity for systems at scale. Therefore, SiP OCSs are also inserted between the ToR and the aggregation layers. When a partial server regrouping is performed, bandwidth steering will be applied to reduce contentions at higher layers. An example is shown in Fig. 4.1(c). Bandwidth steering above the ToR is used to relocate connections from the ToR to desired aggregation EPSs. The overall system architecture essentially reconstructs locality of connection and optimizes topology to better fit network traffic demands.

4.2.2 SiP Switches and Control

Our proposed architecture and control scheme are agnostic to the choice of SiP switching devices. We optimized our control scheme of the SiP switches and fabricated custom DAC cards to provide a software-based network control interface, and to ease control implementation complexity and achieve better integration. Depending on the traffic patterns of the distributed deep

learning training, the overall network controller reconfigures network topology on demand. Figure. 4.2(a) shows our overall network control plane. It consists of (1) a Ryu-based SDN controller that manages the flow tables on the EPSs; (2) a TCP/IP client program that sends new reconfiguration requests to the SiP OCS subsystem as shown in Fig. 4.2(b). In the subsystem, the SiP network controller is built upon a Xilinx ZCU 106 board. We leverage Xilinx PetaLinux to build a kernel image stored in a SD card and boot a Linux/Ubuntu operating system (OS) from a hard-drive. A TCP/IP server program running on the ARM processors responds to the reconfiguration requests from the overall network controller. Control algorithms such as calibration and thermal stabilization can be not only implemented in the software for simplicity, but also implemented as hardware logic in the field programmable gate array (FPGA) for control speed. A custom 80-channel DAC daughter card was fabricated to demonstrate a path toward large-scale system integration. Using the DAC daughter cards, the switch controller implemented on a Terasic TR4 board provides correct bias voltages to the switching elements in the packaged SiP switches. The SiP network and switch controllers are connected using GPIOs and the interface from SiP switch controller to the fan-out PCB uses SMA cables. Figure. 4.2(c) shows the physical devices.

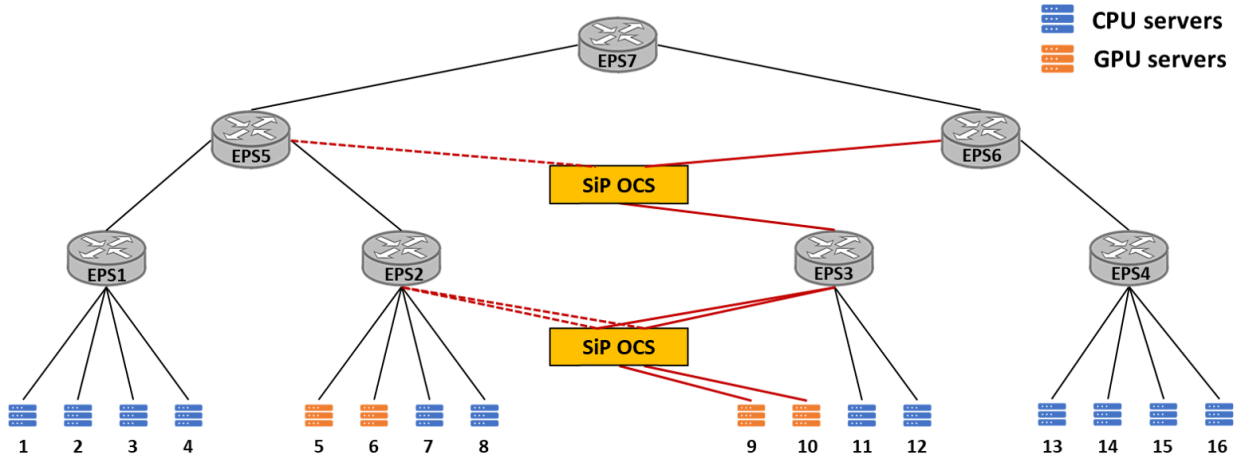


Figure 4.3: A 16-node experimental testbed with SiP OCSs and EPSs in a reconfigurable fat tree topology.

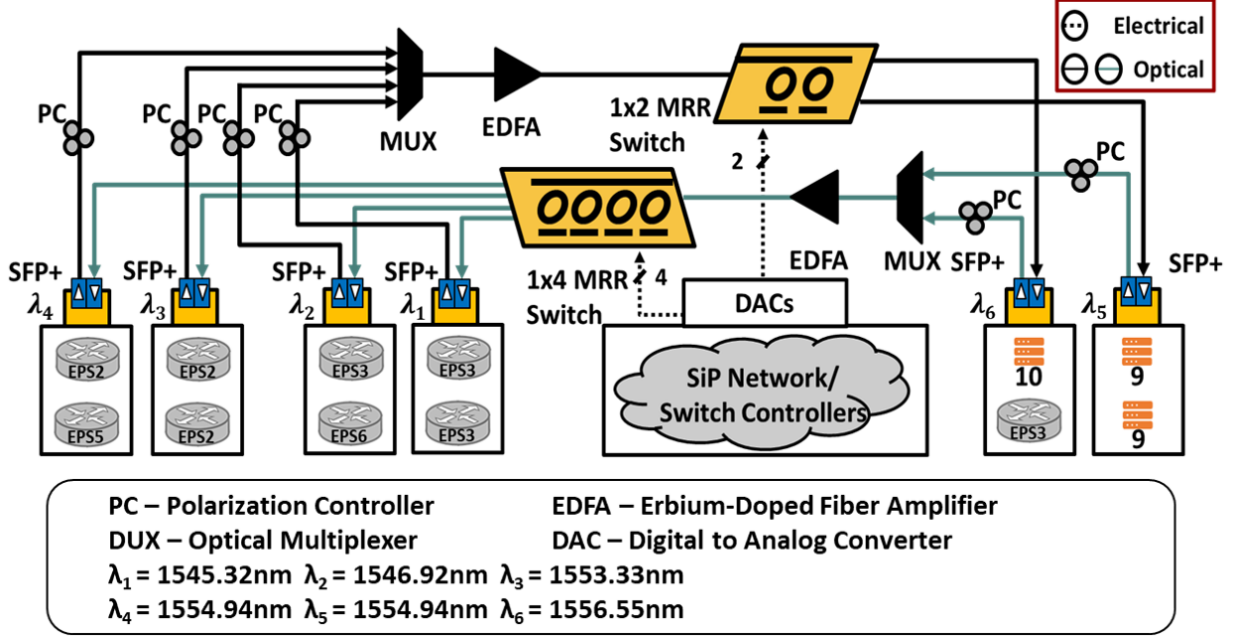


Figure 4.4: Experimental setup demonstrating the cases of server regrouping and bandwidth steering above the ToR.

4.3 Testbed

To run distributed deep learning workloads and demonstrate the network improvements of our proposed system architecture, we built a 16-node HPC/datacenter testbed as shown in Fig. 4.3. We used 4 GPU servers (in orange) equipped with NVIDIA M40 GPU to run ring all-reduce and parameter server training algorithms across them. The other 12 CPU servers (in blue) are used for running other applications to generate background traffic across the network. The EPSs are virtually partitioned from an OpenFlow-enabled PICA8 packet switch with 10G SFP+ ports. We use a 1×2 and a 1×4 MRR-based OCSs to perform server regrouping and bandwidth steering above the ToR EPSs. For the fully server-regrouped case (defined as case #1), the SiP switches are connected to servers #9 and #10, and two separate ports on EPS #2 and #3, respectively. For the case (defined as case #2) where the server regrouping is partially performed and bandwidth is steered above the ToR, the SiP switches are connected to server #9 and a port on EPS #3, and individual ports on EPS #3, #6, #2, #5 respectively. In this case, server #10 is connected to EPS

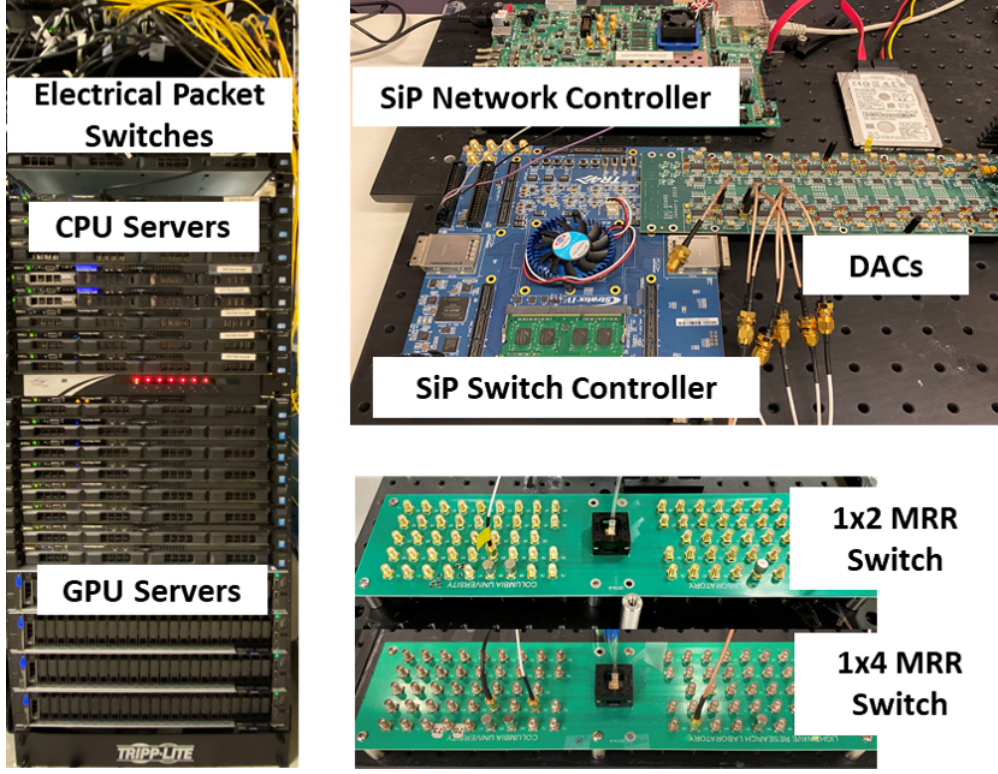


Figure 4.5: A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller.

#3 without going through the SiP OCSs. 10G SFP+ optical transceivers are used for reconfigured links and static links are using 10G electrical transceivers.

A detailed experimental setup is shown in Fig. 4.4. Two SFP+ transceivers with wavelengths at 1554.94 nm (λ_5) and 1556.55 nm (λ_6) are used for server #9 and #10 to transmit data to EPS #3 and EPS #2 (in case #1) or for server #9 and EPS #3 to transmit data to EPS #3, #6, and EPS #2, #5 (in case #2). Four SFP+ transceivers, with wavelengths at 1545.32 nm (λ_1), 1546.92 nm (λ_2), 1553.33 nm (λ_3) and 1554.94 nm (λ_4), are used for the opposite direction. The polarization controllers (PC) are used to maximize the optical power being coupled into and out of the SiP chips. An erbium doped fiber amplifier (EDFA) is necessary to compensate the loss due to the grating couplers of the SiP switch chips. We note that the approaches described in Section II can potentially reduce the loss and allow the system to work without EDFAs. Detailed SiP switching characteristics can be found in the previous work [116]. The SiP network controller FPGA board (ZCU 106) receives

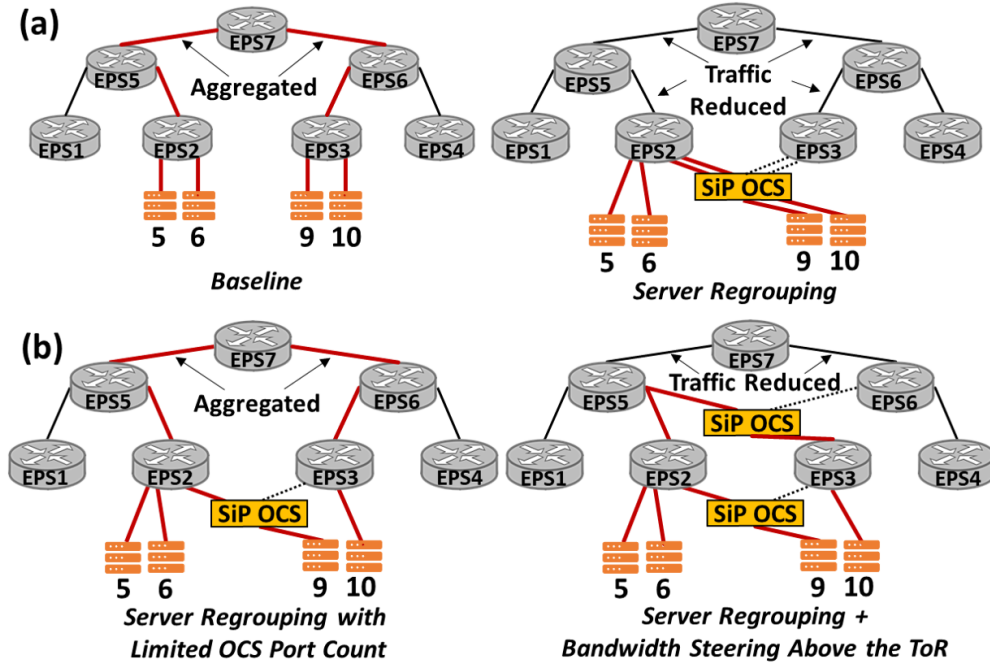


Figure 4.6: A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller.

configuration requests from the overall network controller and triggers the SiP switch controller (TR4). The switch controller will then configure each MRR by tuning the resonance with bias voltage. A photograph of EPSs, CPU servers, GPU servers, SiP switches, SiP network controller, and SiP switch controller is shown in Fig. 4.5.

We note that the reconfiguration speed limitation is the transceiver locking and the EPS polling time⁴⁹ at the millisecond scale. This is a negligible effect in the current architecture due to the fact that the topology reconfiguration only happens before an application starts. Thermal drift of the MRR-based switches could lead to system performance degradation, and thermal stabilization [58, 59] should be applied to address this issue before the deployment of MRR-based architectures in future datacenter/HPC networks. The experiments described in this work take place in a thermally stable environment.

4.4 Experiments and Results

We used the distributed communication package in PyTorch,⁵⁰ which enables the processing groups for each of the workers used in the synchronized training and the parameter server and workers in the asynchronous training. The training jobs run across 4 server nodes (#5, #6, #9, #10) for the ring all-reduce algorithm and run across 3 server nodes (#5 – parameter server and #9, #10 - workers) for the parameter server algorithm. For the remaining 12 servers, we run skeletonized version of the Gyrokinetic Toroidal Code (GTC) benchmark applications [117] as the background traffic across the network. There are two test cases. (1) Assuming OCS port count is sufficient for server regrouping, we use baseline (no dynamic reconfigured links) to compare with server-regrouping (servers #9, #10 regrouped to EPS #2) as our test case #1. (2) For test case #2, a partial server-regrouping (only server #9 regrouped to EPS #2) compares server-regrouping with bandwidth steering above the ToR (server #9 regrouped to EPS #2 and a steered link from EPS #3 to EPS #5). Simplified diagrams for the two test cases are shown in the Fig. 4.6(a) and (b), respectively.

Figures 4.7, 4.8, 4.9, and 4.10 plot the throughput of incoming traffic to servers #9 and #10 (blue and red), from EPS #5 to EPS #7 (green), and from EPS #1 to EPS #5 (yellow) for various training strategies and test cases. The plotted links are sufficient to show the network performance of deep learning workloads. The neural network is VGG1 for image classification and the dataset is imagenette [118]. Figures 4.7, and 4.8 show the results for the synchronized training. For test case #1, Fig. 4.7, the green curve in the baseline diagram (top left) indicates the traffic at the core level is aggregated by the background GTC traffic (yellow) and the ring all-reduce training traffic (red or blue). The training process is suppressed by the background GTC traffic, and it takes approximately 5341 s to train the VGG network for 1 epoch for the baseline. For the regrouped case, server #9 and server #10 are regrouped to EPS #2, and the training job's traffic is within EPS #2, such that the communication bandwidth for the ring all-reduce training processes is restored. The red curve in the server regrouping diagram in Fig. 4.7 shows a 5 Gb/s bandwidth on average for

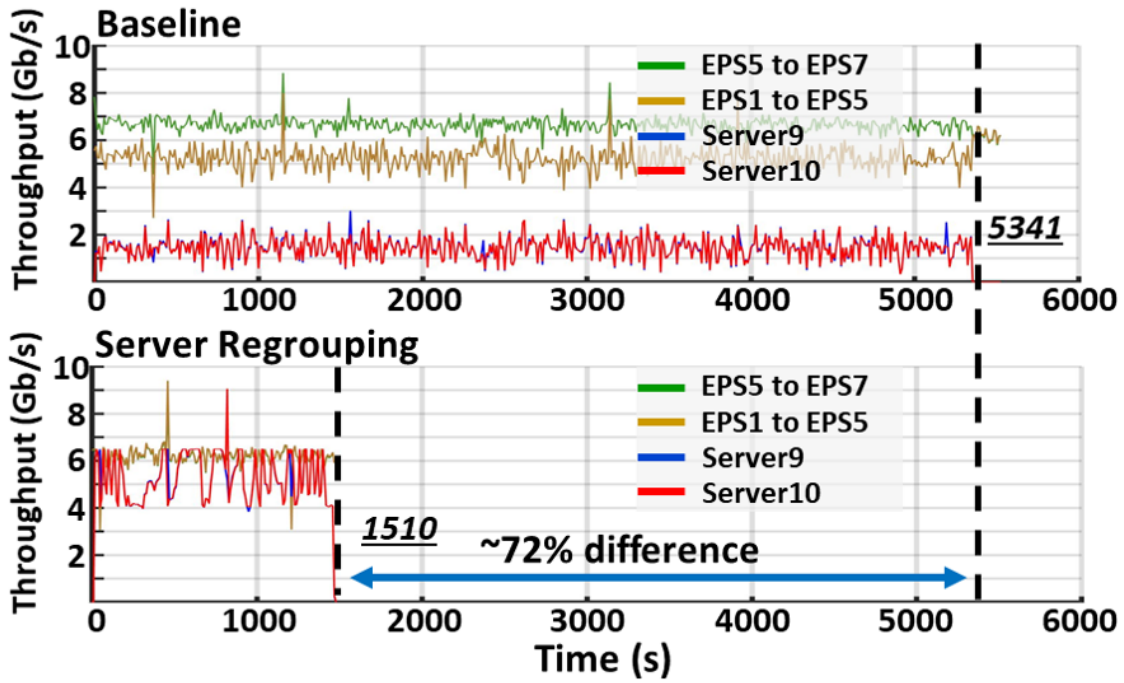


Figure 4.7: Throughput of the links to server #9 and #10, from EPS #1 to EPS #5, and from EPS #5 to EPS #7 for test case #1 in the synchronized training of the VGG neural network.

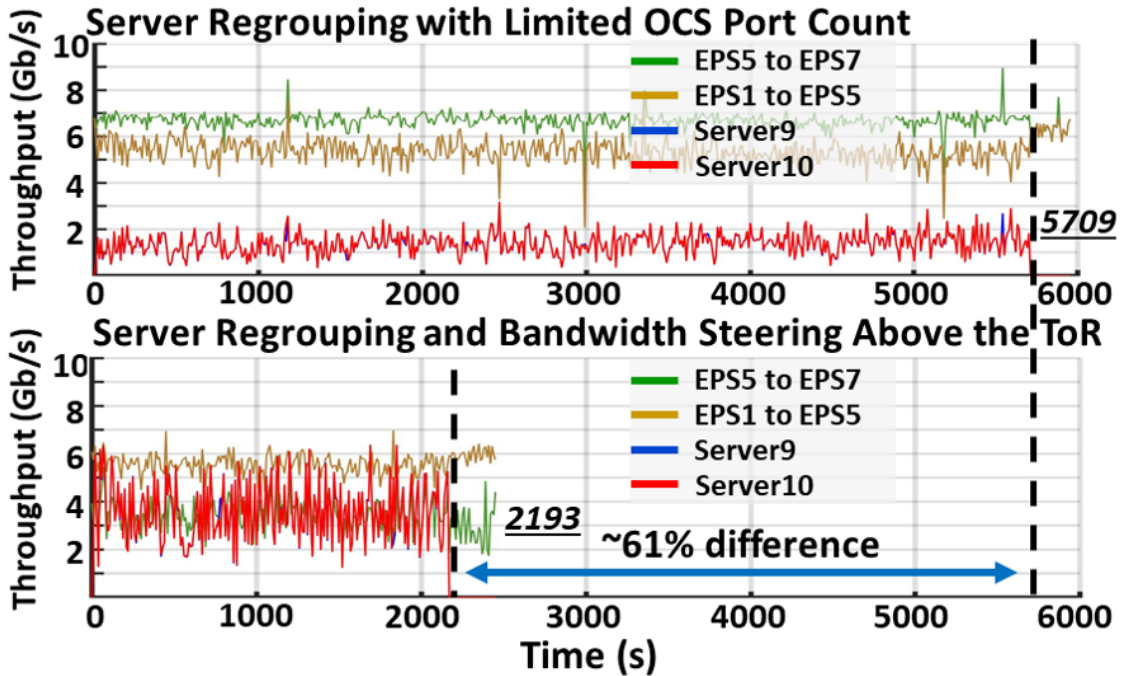


Figure 4.8: Throughput of the links for the test case #2 in the synchronized training.

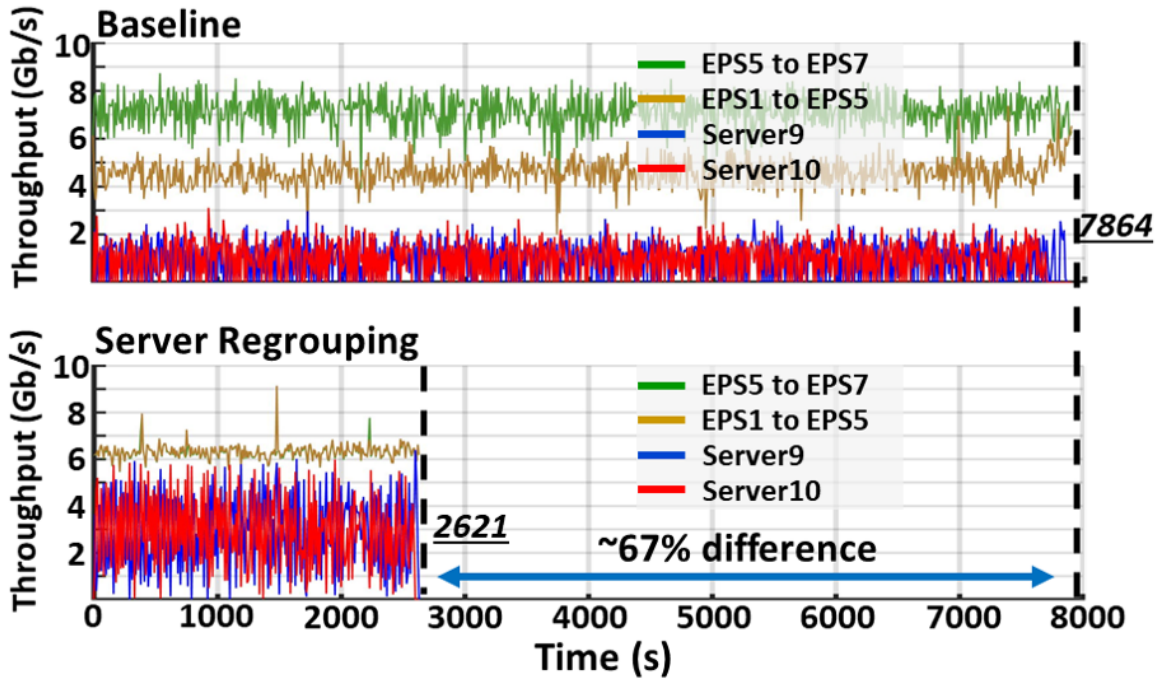


Figure 4.9: Throughput of the links for the test case #1 in the asynchronized training.

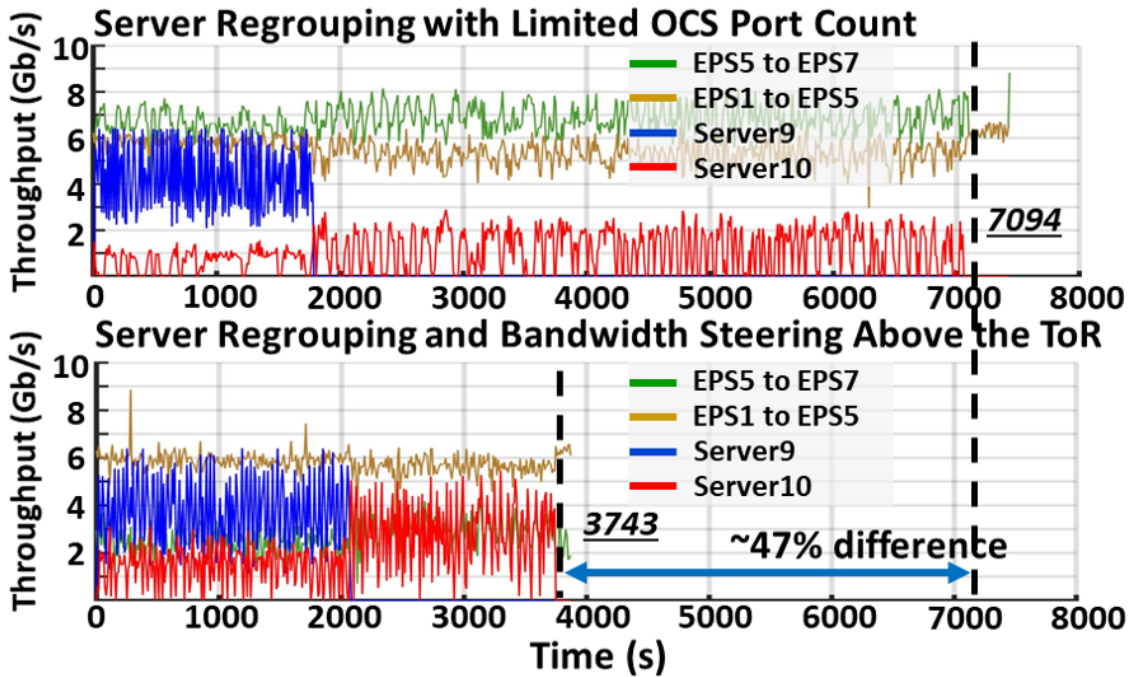


Figure 4.10: Throughput of the links for the test case #2 in the asynchronized training.

the ring all-reduce algorithm with a 72% difference in execution time which corresponds to a 3.6 \times network performance improvement. For test case #2, Fig. 4.8 right shows the results for server regrouping with limited OCS port count and when bandwidth steering above the ToR is applied. We observe a 5709 s execution time (in 4.8 top) when only server #9 is regrouped to EPS #2 and no bandwidth is steered between ToR and aggregation EPSs. In comparison, server regrouping and bandwidth steering above the ToR (Fig. 4.8 bottom) provides a 61% difference in execution time which corresponds to a 2.6 \times network performance improvement due to the fact that the deep learning training flows are not going through the core layer of the network. Figure 4.9 shows the performance improvements for the parameter server training algorithm. Similar performance improvements are observed for the server regrouping and the server regrouping with bandwidth steering above the ToR as 67% and 47% in execution time differences (3.0 \times and 1.9 \times improvements), respectively. We should note that parameter server training is an asynchronized training, and it is reasonable that the two worker nodes finish their individual training job at different time stamps as indicated by the red and blue curves in Fig. 4.10. The comparative experimental results can be found in Table 4.1.

4.5 System-scale Evaluation

We study the scalability and network performance of the proposed system architecture on two distributed deep learning training algorithms: (1) ring all-reduce and (2) parameter server. For each of the workloads, we analyze how server regrouping and bandwidth steering affect the performance of large-scale networks with various tapering ratios. In addition to using uniformly mapped jobs as a performance upper bound, we also simulate non-uniformly mapped jobs since past work [113] has shown that frequent system fragmentations in high performance systems could make the job mapping largely non-uniform. For the purpose of this work, which is to show the performance improvement of the proposed strategies, we assume that server regrouping and bandwidth steering strategies for non-uniform job placements happen before a workload starts in the simulation and therefore has no packet loss due to reconfiguration. We plan to add this reconfiguration function-

load-balancing.

We test server regrouping on both ring all-reduce and parameter server types of traffic workload in our simulation. The ring all-reduce traffic and parameter-server traffic each contains 32 and 16 compute nodes per job, respectively. Under uniform job placement, each process is mapped sequentially onto each server. However, job placement might not always be uniform in real high performance systems. Past work [113] has shown that applications are often placed on a set of distant and non-contiguous nodes, resulting in system fragmentation. In order to verify the effectiveness of server regrouping at large scale, we introduce a mixed job mapping strategy to generate a more adversarial traffic pattern to the static fat tree. This mapping hinges on the ratio of intrapod to interpod traffic. For our simulation, we set the ratio so that half of the nodes in each pod communicates with other nodes in the same pod, and the other half would communicate with nodes in the other pods. The mapping within each half is also shuffled to introduce randomness in the mapping.

4.5.2 Server Regrouping and Bandwidth Steering

Since the usage of small-radix OCSs could impose extra physical constraints on the topology-wiring problem [120], we first make the assumption that given k ToRs with k downlinks each, the OCS layer in between the server and the ToR layer is comprised of a single large-radix OCS with k^2 ports. The OCS can be viewed as a k^2 by k^2 fully non-blocking switch capable of reaching and regrouping servers across all pods. This assumption is necessary for the purpose of our work which is to evaluate the performance and scalability of server regrouping strategy. Experiment with this initial assumption can serve as a performance upper bound for our experiments with smaller radix OCSs. To evaluate the network performance of smaller-radix OCSs, we split the single large-radix OCS into two OCSs with half the radix (each connecting two pods) and into four OCSs with a quarter of its original radix (each connecting one pod). The server regrouping heuristic for the ring all-reduce traffic is similar to that of the parameter-server, and it can be split into two substeps:

1. Group jobs that only contain servers in the same pod. For these jobs, group as many servers

under the same ToR switch as possible.

2. Group jobs that contain servers in different pods. If the OCS port count is big enough to reach all the servers in the same job, then group these servers under the first available ToR. If not, group as many servers in the same job as possible under a single ToR for each pod that contains these jobs.

When server regrouping falls short, bandwidth steering at the ToR to aggregation layer can be applied together with server regrouping to further improve the performance. We assume a single large OCS between the aggregation and core layers as a performance upper bound. Smaller radix OCS could also be used here similar to previous works^{13,21} for bandwidth steering at higher layers depending on the reconfiguration requirements and tapering ratios. Bandwidth steering above the ToRs is configured so that the number of flows traversing the core layer is minimized. This is done by first regrouping the servers that have the same destination pod under the same ToR switch within each pod and then wire the OCS such that the two ToRs with the heaviest communication are connected by the same aggregation switch. For these simulations, we assume that the topology is reconfigured according to the described server regrouping or bandwidth steering strategy before a workload starts.

4.5.3 Results

In this section we evaluate the performance of server re- grouping and higher-layer bandwidth steering in large-scale systems. Figure [fig:ddlSimulationResult] shows the simulation results for average flow throughput (for both intrapod and interpod flows) as the job- mapping and topology design vary for all ToR-to-aggregation tapering ratios. Uniform job placement corresponds to the case where jobs are mapped sequentially onto the servers in the topology without any shuffling. It achieves the highest average throughput since the communicating nodes are placed close to each other, so the tapered core layer links are not congested by interpod flows. It serves therefore as the performance upper bound for all the server regrouping schemes in our experiments. Indeed, when servers are regrouped with one large-radix OCS (RG (#OCS = 1)), results for both ring all-

reduce workload in Fig. 4.12(a) and parameter server workload in Fig. 4.12(b) show matching behavior with the uniform case. Note here that the mean flow throughput for parameter server is much lower than that of ring all-reduce because parameter server jobs contain many more flows in each iteration, resulting in much higher link congestion. On the other end, baseline corresponds to the case where jobs are randomly mapped onto servers across different pods without server regrouping.

RG (#OCS = 2) and RG (#OCS = 4) correspond to the cases where the servers are regrouped with two and four OCSs respectively. Since we have four pods in total, the former case with two OCSs would mean that each OCS connects to servers and ToRs in two pods. Similarly, the latter case with four OCSs means that each OCS is responsible for connecting servers and ToRs in only one pod. For the cases where servers can only be partially regrouped, we still observe improvement from the baseline case, especially under higher tapering. Although not all servers can be regrouped, other properly re-grouped servers have already reduced the amount of traffic traversing the tapered layer by an appreciable amount.

On top of server regrouping, we can further improve the performance of the network by employing bandwidth steering in the ToR-to-aggregation layer to alleviate congestion at the top layers. We see that for the cases with both server regrouping and higher-layer bandwidth steering (BS), the performance is consistently higher than the purely regrouped cases, especially at lower tapering ratio. For higher tapering ratios, the number of available reconfigurable links between each ToR switch and the aggregation switches is limited, which limits the benefits of higher-layer bandwidth steering.

Simulation results show that our approach improves the network performance for the ring all-reduce and parameter server workloads at scale. We only consider results with two or more OCSs as the RG (#OCS = 1) case is the same as our performance upper bound. We found that for 2× tapering ratio, server regrouping alone can improve the throughput performance from the baseline by 2.3×, and higher-layer bandwidth steering can provide up to 11% further improvement (2.5× improvement in total from the grey bar to the pink bar). For higher tapering ratios, the total im-

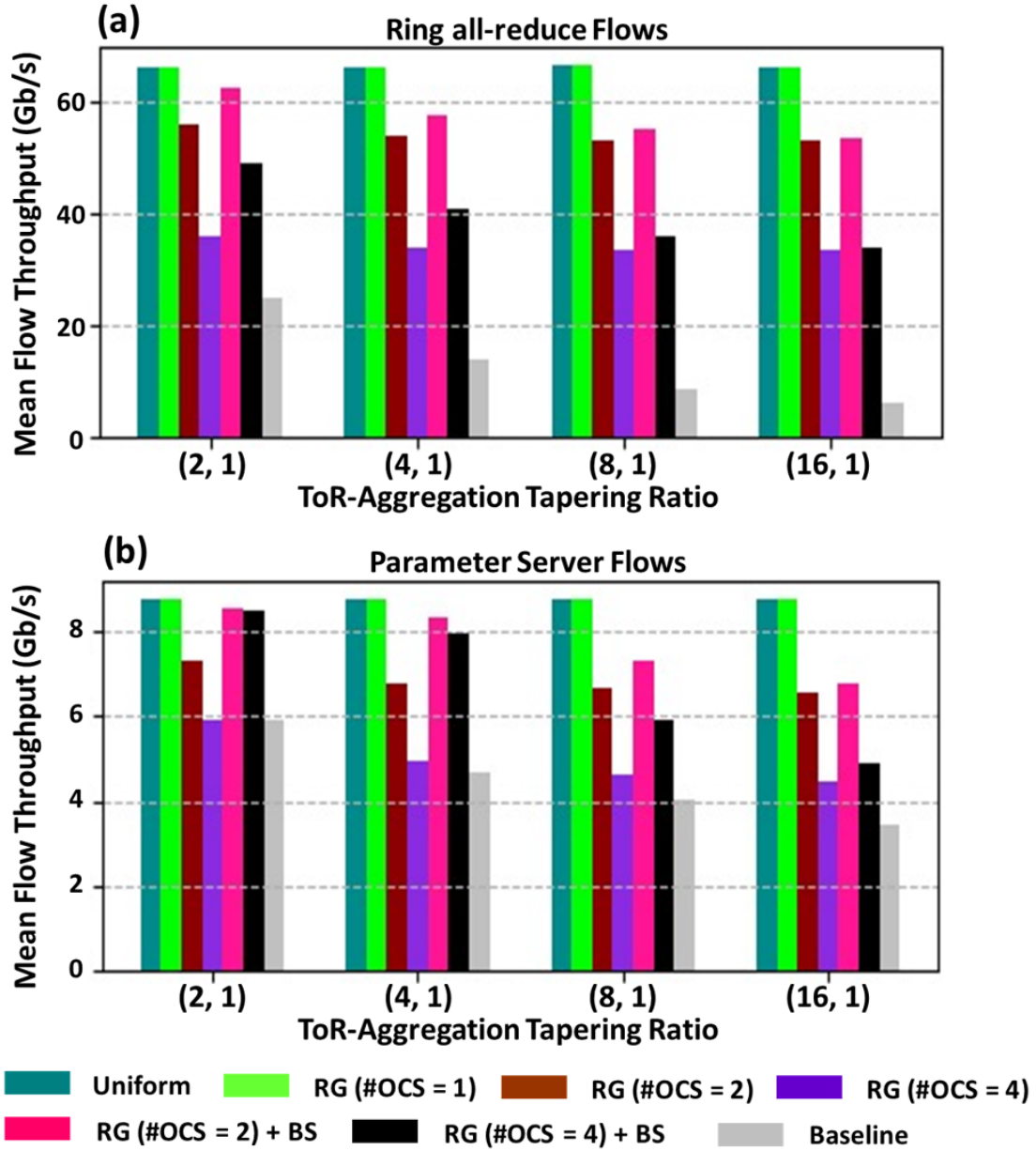


Figure 4.12: Average flow throughput of all the flows as a function of the tapering ratio for all the traffic and job mapping scenarios. RG denotes regrouping and BS denotes higher-layer bandwidth steering. With the exception of Uniform (uniform job-mapping), all other cases assume an adversarial interpod job mapping as described in the Simulation Setup Section. (a) Results for ring all-reduce flows. (b) Results for parameter server flows.

provements can reach up to $8.6\times$ for ring all-reduce and $2\times$ for parameter server, respectively. Table 4.1 shows the performance improvements of different architectures including both experiments and simulations.

Table 4.1: Experiment and Simulation Performance Measurements

Configurations	Improvements
Server regrouping compared to baseline for ring all-reduce training (experiment)	$3.6\times$
Server regrouping with bandwidth steering compared to server regrouping with limited point count for ring all-reduce training (experiment)	$2.6\times$
Server regrouping compared to baseline for parameter server training (experiment)	$3.0\times$
Server regrouping with bandwidth steering compared to server regrouping with limited point count for parameter server training (experiment)	$1.9\times$
Server regrouping using two OCSs on $2\times$ tapered fat tree compared to baseline for ring all-reduce training (simulation)	$2.3\times$
Server regrouping using two OCSs with bandwidth steering on $2\times$ tapered fat tree compared to baseline for ring all-reduce training (simulation)	$2.5\times$
Server regrouping using two OCSs on $2\times$ tapered fat tree compared to baseline for parameter server training (simulation)	$1.2\times$
Server regrouping using two OCSs with bandwidth steering on $2\times$ tapered fat tree compared to baseline for parameter server training (simulation)	$1.4\times$

4.6 Chapter Summary

In this chapter, we have shown a reconfigurable datacenter/HPC system architecture using SiP switches to accelerate distributed deep learning training workloads. We used VGG as the primary workload for our experiment, but our proposed architecture could work on a wide range of distributed machine learning applications that employ ring all-reduce or parameter-server types of collective operations. Using silicon photonic switches, we introduce topological reconfigurability at two network levels to achieve two optimization goals: (1) server-regrouping by introducing SiP OCSs between the ToR switches and servers, and (2) bandwidth-steering by introducing SiP OCSs between the ToR and aggregation layers. We demonstrate our proposed architecture using a physical testbed with 16 nodes arranged in a fat tree topology and show up to $3.6\times$ network improvement. At system scale, server regrouping delivers a $2.3\times$ flow throughput improvement and higher-layer bandwidth steering provides a further 11% improvement in a $2\times$ tapered fat tree. These results show the proof-of-concept functionalities of our proposed system architecture and the potential of integrating SiP switches in datacenters and HPCs to improve DL training performance.

Chapter 5: Disaggregated Architectures for Deep Learning

5.1 Flexible Optically Interconnected Computing Resources

5.1.1 Introduction

Distributed machine learning applications have become a significant part of today’s high performance computing and data center workloads [121]. When training a distributed machine learning model across multiple nodes due to system fragmentation in the conventional server-centric architecture, inter-node communication becomes more bottlenecked than intra-node communication due to both higher latency and lower link bandwidth. The problem becomes even worse when scaling to a larger number of nodes as the inter-node links will be further congested due to heavy inter-node communication. This results in significant resource under-utilization in conventional server-centric architectures [122]. Resource disaggregation provides a solid solution to this problem by providing a flexible allocation of fine-grained hardware into virtual nodes [88]. Optical circuit switches (OCSs) provide high bandwidth, low cost solution to de-fragment and pool disaggregated resources with minimal latency overhead. In previous work, OCSs have been proposed to achieve resource disaggregation [122, 123, 124], however, the prior work was limited to a generic concept without system-level demonstration.

In this section, we demonstrate rack-scale GPU disaggregation with OCSs and emulated aggregator switches, and show the capability to reconfigure resources to reduce communication overhead. Our proposed architecture leverages the flexibility of optical switches to increase hardware utilization and reduce application runtimes by alleviating system fragmentation. We build up our testbed using 4 commercial rack servers with 6 GPUs, 2 emulated aggregator switches and a MEMS switch to demonstrate resource disaggregation and defragmentation using application containers. Our testbed experimental results show 73.2% and 62.0% improved workload comple-

tion time with 3x increased GPU utilization in two test cases.

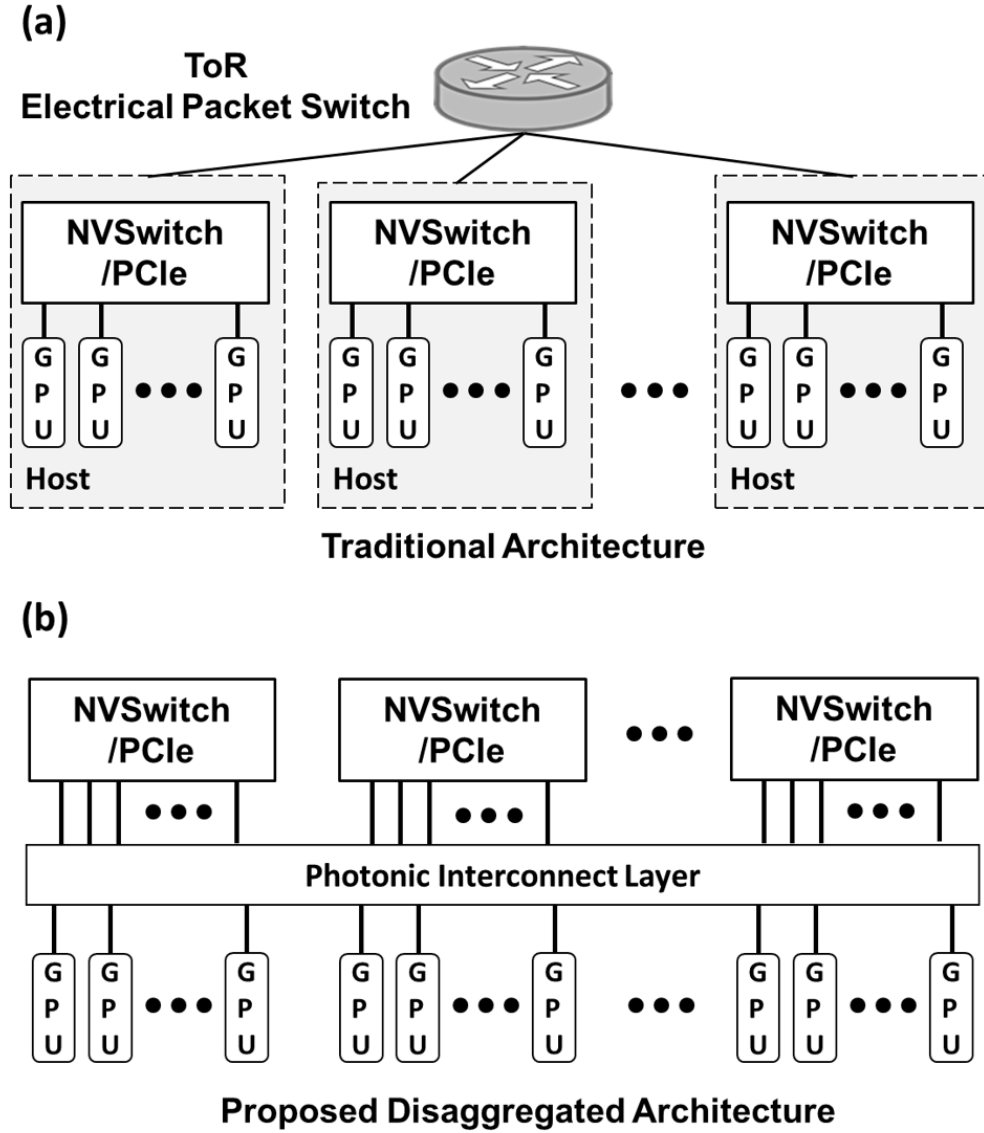


Figure 5.1: (a) Traditional architecture for GPU clusters. (b) Proposed system architecture for disaggregated GPU resources.

5.1.2 System Architecture

Figure 5.1 shows our proposed system architecture with Aggregator Switches (AS), Optical Circuit Switches (OCS) and a pool of disaggregated GPU resources. The aggregator switches are essentially packet switches that connect a group of heterogeneous resources. For example, an

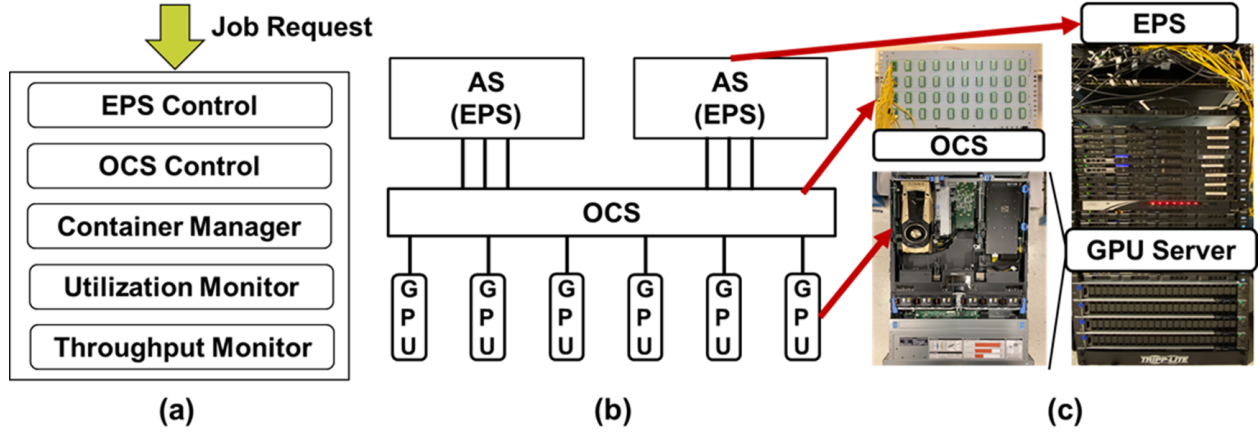


Figure 5.2: (a) Network control plane. (b) Testbed architecture. (c) Hardware implementations using Calient MEMS switch, server rack with ToR switch and rack server.

equivalent device in traditional architecture would be NVSwitch or PCIe Switch. Traditionally in distributed AI/ML applications, nonstop synchronization of large gradients implies an urgent need to allocating its GPU instances in a single machine. Top-of-rack switches can act as aggregators to hold recomposed resources and to connect multiple types of devices. By reconfiguring the optical links between GPUs and our aggregator switch, disaggregated resources can be rearranged to reduce system fragmentation and increase job locality. As a result, this leads to cost and performance benefits of the disaggregated system from increased utilization and maximized traffic locality. Nevertheless, accommodating OCSs in the network of disaggregated resources enables changing the network topology as needed by the application. For GPUs, different AI/ML applications have different requirements on the number of GPUs needed, varying from one to hundreds. Furthermore, this architecture can be adapted to provide flexibility to re-assemble heterogeneous computing resources for specific application needs, such as CPU, memory. In this case, the issue of low port counts of OCSs can be tackled by using multiple low-radix switches at the same time combined with other methods like locality-driven job scheduling.

5.1.3 Experimental Setup and Results

We build up our testbed with 6 NVIDIA GPUs installed into 4 commercial rack servers to demonstrate the capabilities of emulating deep disaggregation of GPUs and resource reconfiguration with OCSs. Each server is installed with a Tesla M40 GPU. In addition, two Titan V GPUs are inserted into the servers. All of them are connected with Mellanox ConnectX-4 NICs that enables RDMA over Converged Ethernet v2 (RoCEv2) for improved performance. In Fig. 5.2 we show the network control plane for our testbed. We use a SDN controller to host management applications, including EPS flow table control, optical switch control, and container management through Ethernet.

When running an ML workload we reserve a certain number of GPUs and create one Docker container for each of them on their corresponding . On the servers, each container has one and only one GPU visible to it. The container’s network I/O is then mapped to a specific NIC port on the physical machine with a unique IP address. By doing this we’re able to (1) send gradient updates directly from one GPU to another without the participation of other resources; (2) assign specified IP address for each GPU resource in the network. This allows us to emulate and study these GPUs as they are deeply disaggregated. We use Tensorflow to build distributed machine learning applications to synchronously train a modified VGG16 neural network across multiple workers. Two different cases of ML workloads were studied on this topology. One in Fig. 3a shows two different jobs (red/blue) requiring 4 and 2 GPU resources for synchronous training, using ring all-reduce algorithm. The other case, as shown in Fig. 3b, shows two workloads with 3 and 3 GPUs required for synchronous training, using the same ring all-reduce algorithm. We obtained network I/O throughput for each Docker container using Ryu SDN monitoring program, and the GPU performance data is queried by nvidia-smi.

In case #1 the red workload spent 407s to complete when there is no reconfiguration, compared to 109s after regrouping all 4 GPUs under the same ToR switch. This yields a 73.2% reduction in completion time and a 3x utilization increase during training phase. In case #2 our workload of concern spent 368s before reconfiguration and 140s afterwards. Here we obtained 62.0% reduction

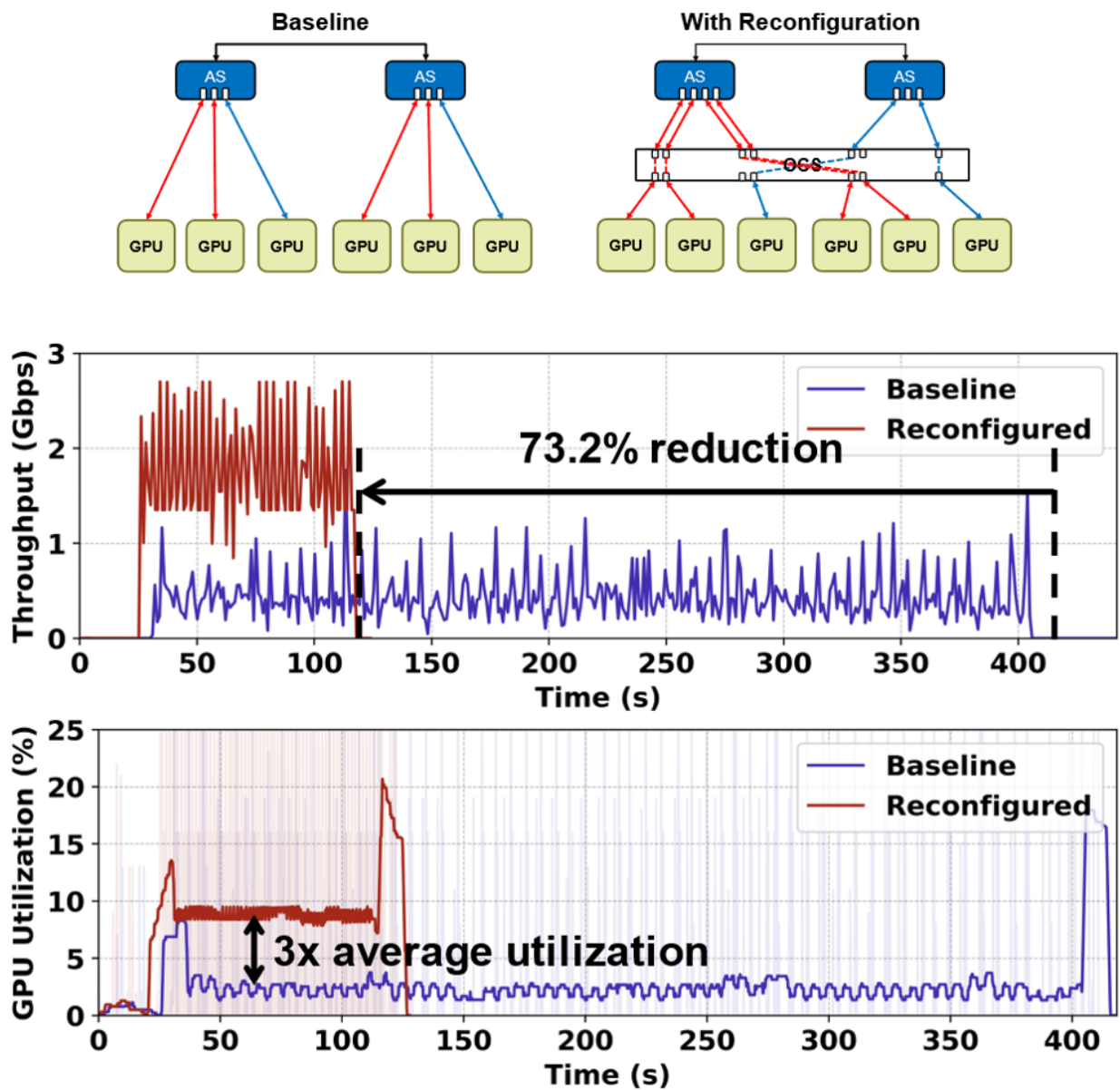


Figure 5.3: Testbed topology for two synchronous ML workload using 4 GPUs and 2 GPUs for training. Experimental results that show network performance and utilization improvements.

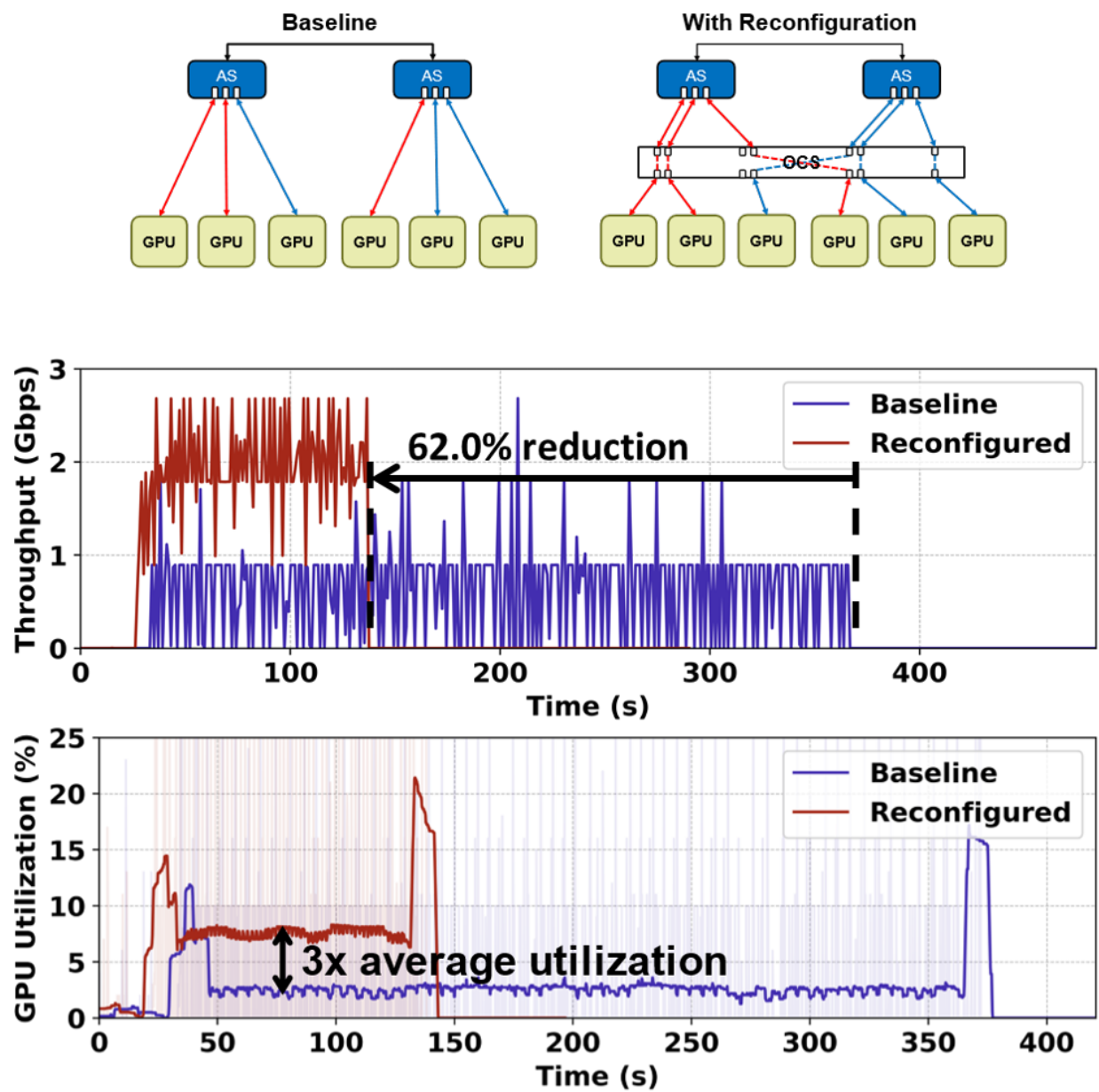


Figure 5.4: Testbed topology for two synchronous ML workload using 3 GPUs and 3 GPUs for training. Experimental results that show network performance and utilization improvements.

in training time, while no significant improvement in resource utilization is observed. However, there is an expected increase in network throughput that indicates a higher effective utilization of the GPU after reconfiguration. Without the reconfiguration, resource fragmentation undermines the application performance of the workload running on top and interferes with other application flows. We find that OCSes can significantly help increase performance and hardware utilization in disaggregated systems running AI/ML workloads.

5.1.4 Conclusion

We demonstrated resource reconfiguration in disaggregated systems for machine learning applications using OCSes. Our proposed control plane separates GPUs in a single server and allow for disaggregation using RoCEv2. With optical switching we showed 73.2% and 62.0% improved training time and $3\times$ increase in GPU utilization. In the future, we will explore implementing our architecture at larger scale.

5.2 SiPAC: Silicon Photonic Accelerated Compute Cluster

5.2.1 Introduction

Distributed deep learning (DDL) has been proposed as a solution to perform large-scale training tasks that require multiple computing units (CU), such as GPUs, TPUs, and other accelerators, to complete. Many specialized hardware accelerators have been proposed to accelerate DDL. Some commercially available examples include Nvidia’s DGX system [125] and Google’s Cloud TPU system [126]. Large-scale models have been reported to be trained on these systems (e.g. Megatron-LM was trained on 3072 Nvidia A100 GPUs [29]). However, as the scale of these models continue to increase, researchers have found that the current systems may face severe communication bottlenecks [29, 127]. Benchmark from the Sierra supercomputer [128] reported DDL communication overhead to be more than 10x the computation time when DDL workload is trained on more than 256 GPUs [129].

Another emerging trend is to incorporate SiP technologies in the network design as SiP interfaces are critical for achieving extreme high bandwidth (Tbps) interconnects. An example of the silicon photonic interface is TeraPHY [130] that supports up to 2 Tbps bandwidth per chiplet. Recently, a promising Kerr frequency comb-driven silicon photonic transceiver architecture [69] has been reported. It leverages a frequency comb [70] for a dense wavelength division multiplexing (DWDM) light source, and uses (de-)interleavers to split and combine wavelength channels in order to scale up the data transmission bandwidth within a single fiber. However, works that proposed using silicon photonic technologies for DDL training [112, 131, 132, 133] have put more emphasis on designing networks using Optical Circuit Switches (OCS) to dynamically reconfigure the network topology to cater for different DDL traffic demands. For example, SiP-OCS [132] co-designs the model partitioning and device placement with a specialized network architecture that employs a layer of reconfigurable OCS. And there is little known work about leveraging the inherent advantages of extreme high bandwidth silicon photonic interconnects, including both transceivers and switches, to achieve efficient communication and architecture design, which will be a main focus of this study. The major contributions of our work are as follows:

Hierarchical All-to-All Interconnection: High bandwidth all-to-all topologies are ideal for network interconnections but are expensive to build due to the high component counts required. To achieve the benefit of the all-to-all topology while saving component count and cost, we propose the SiPAC topology which is an optical switched topology that leverages multi-wavelength selective switches and high-bandwidth dense wavelength division multiplexing (DWDM) links to build a hierarchical all-to-all topology with a physical topology based on the BCube topology [134]. This architecture ensures low network diameter while providing high-bandwidth direct paths for collective communications (e.g. all-reduce, all-to-all) that are commonly used in DDL workloads.

Multi-Wavelength Selective Crossbar MRR Switch: We design a micro-ring resonator (MRR) based multi-wavelength selective crossbar switch (WSS) that exploits the periodic property of the free spectral range (FSR) of the MRRs. By carefully engineering the FSRs, we demonstrate the ability for each MRR to drop multiple wavelengths, thus increasing the effective

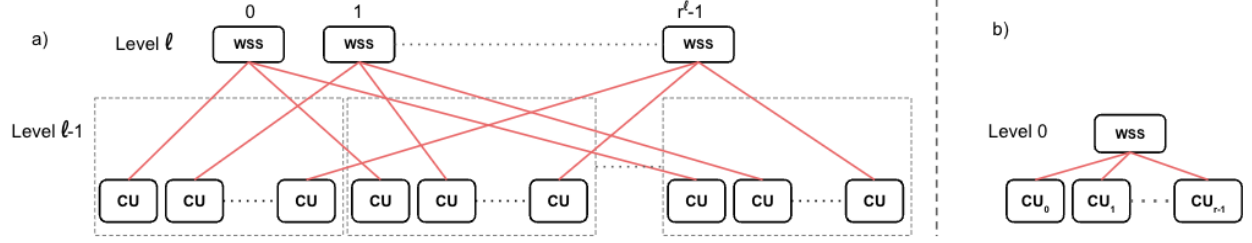


Figure 5.5: a) SiPAC architecture based on the recursive BCube topology with $L = l + 1$ levels. The l th level is constructed from r^l r -port switches and $r(l - 1)$ th level units. b) The base unit of the SiPAC topology where r CUs are connected to a WSS.

bandwidth per ring. The cascaded ring structure separates the incoming wavelengths into sub-groups and recombine the interleaved wavelengths into common output buses. It shuffles the input wavelengths to different outputs and effectively achieves the optical broadcasting functionality. To demonstrate the feasibility of our architecture, we conducted a testbed experiment where an array of wavelengths are shuffled by a cascaded ring switch, with each ring selecting and forwarding multiple wavelengths.

5.2.2 Topology Design

The SiPAC architecture leverages the multi-wavelength selective property of the MRR crossbar switch to realize a hierarchical full-mesh topology based on the BCube physical topology that has been shown to have low network diameter and high capacity for one-to-x and all-to-all traffic patterns [134]. $\text{BCube}(r, l)$ is a recursively defined, server-centric network topology, where r is the switch radix and l is the level in the topology for $l \in [0, L - 1]$ (notations used are different from the original work) [134] where $L = \max(l) + 1$ is the total number of levels. A base unit BCube_0 is constructed from connecting r servers to an r -port switch. For SiPAC, instead of using servers as endpoints, we replace each endpoint with a disaggregated CU, equipped with l optical transceivers. [insert work that builds on-chip transceiver + citation]. In SiPAC, we also replace each electronic packet switch (EPS) with a multi-wavelength selective optical switch. The rest of the physical topology is constructed similarly to a BCube, replacing each EPS with the wavelength selective optical switches at each level.

Therefore, a general SiPAC_l ($l \geq 1$) of level l is constructed from r SiPAC_{l-1} s and r^l r -port switches, totaling $N = r^{l+1}$ CUs and $l + 1$ levels of switches, as shown in Fig.5.5(a). CUs in a SiPAC_l have $L = \max(l) + 1$ optical ports and are connected to a optical switch in each of the L levels. L is typically small since the number of endpoints grow exponentially as a function of L . For example, using radix-16 WSSes, we could achieve 256 CUs for $L = 2$ and 4096 CUs for $L = 3$. And since the diameter of this topology is also L [134], the resulting SiPAC topology has low diameter. To be more flexible in terms of the number of endpoints, irregular SiPACs can also be built using switches of different radix similar to how partial BCubes are built in [134].

The optical transceiver ports can be directly integrated onto the chip interposers [63] and therefore obviate the need to go through any NIC to reach any other CUs. The total bandwidth of a DWDM link B depends on the number of wavelengths w supported by each transmitter and the per-wavelength bandwidth u , giving $B = wu$. The resulting interconnection network allows for transparent optical switching and therefore achieves direct CU-to-CU communication without any bandwidth difference that appears in commercial GPU clusters. The packet-switch-less design also mitigates packet buffering and avoids any intermediate queuing delays.

The SiPAC architecture enables arbitration-free all-to-all connections for CUs that are connected to the same WSS (across different groups at different levels). This effectively achieves a generalized HyperCube topology [135] with $l + 1$ dimensions, allowing each GPU to communicate directly with $(l + 1)(r - 1)$ other GPUs with reduced link count and transceiver count.

5.2.3 Wavelength Routing/Selection

In SiPAC, each CU is able to communicate with every other CUs that are connected to the same WSS with uniform bandwidth. Since each CU is equipped with the same transmitter, meaning the wavelengths being transmitted from the input ports are the same, we need to tune the MRRs in the crossbar switches so that the wavelengths dropped by different MRRs to the same output bus are different.

We employ a simple round-robin scheme to achieve this wavelength multiplexing. The w

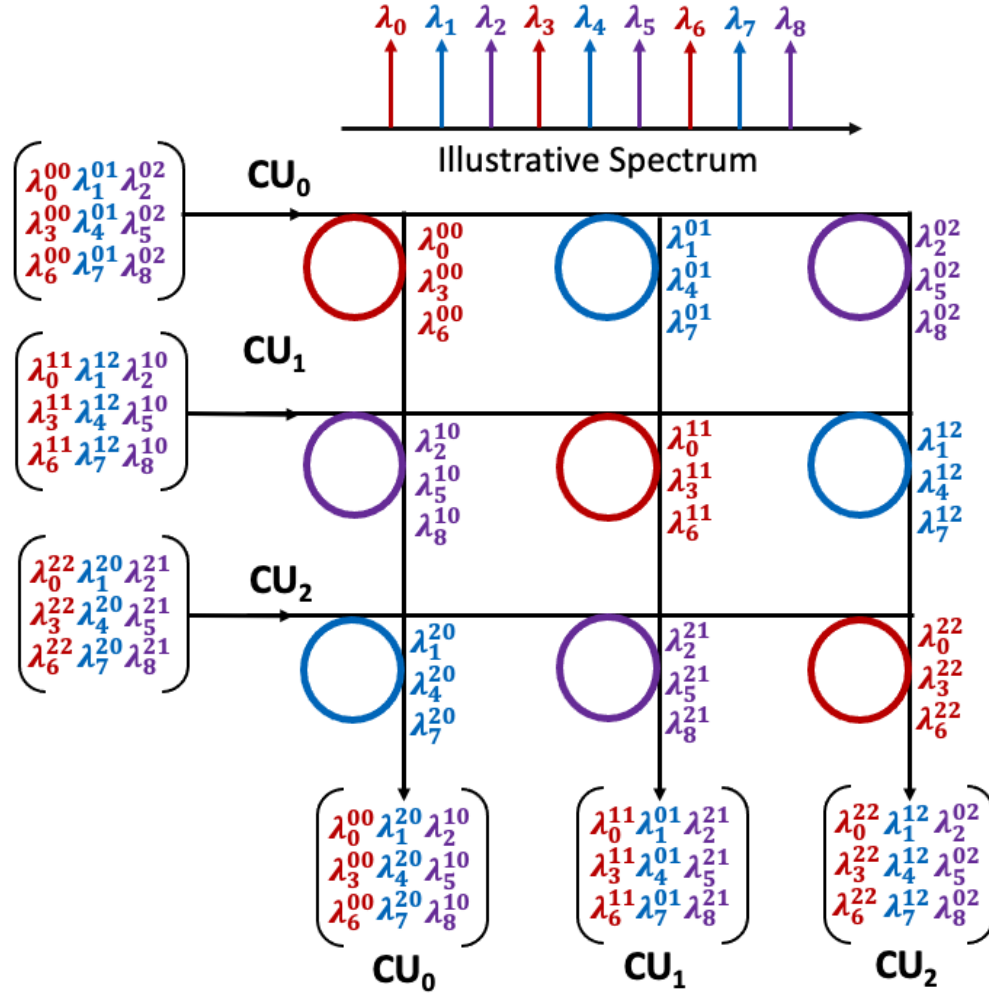


Figure 5.6: Example of wavelength multiplexing for a 3×3 multi-WSS and 8 wavelengths per transmitter ($w = 8$).

wavelengths, $\lambda_k^{i,j}$, $k \in [0, w - 1]$, $i, j \in [0, r - 1]$ being transmitted using an $r \times r$ crossbar switch are divided into r groups. Each group $g \in [0, r - 1]$ contains w/r wavelengths with the wavelength number $k \in [0, w - 1]$ separated by an integer multiple of r . Each group of wavelengths is also labeled with their input port (i) and output port (j). The wavelengths from each input port are interleaved at the crossbar switch in a way so that the drop bus for each output port contains all different wavelengths. This can be done by tuning the rings so that the g th group of wavelengths from input port i are dropped at output port j where $j = (g + i) \bmod(r)$. Fig. 5.6 shows an example of a 3×3 crossbar switch connected to 3 CUs ($r = 3$), with 9 transmitter wavelengths per CU ($w = 9$). Each color represents a different group g of wavelengths (i.e. red = 0, blue = 1, purple

= 2). The colors are interleaved vertically so that each output port gets wavelengths of different colors.

The amount of allocated bandwidth b between each CU pair connected to a common WSS therefore depends on the number of transmitter wavelength w and the number of connections to the WSS in the same level r . For a given bandwidth-per-wavelength u , the CU-to-CU bandwidth is given by $b = wu/r = B/r$.

5.2.4 Testbed Experiment

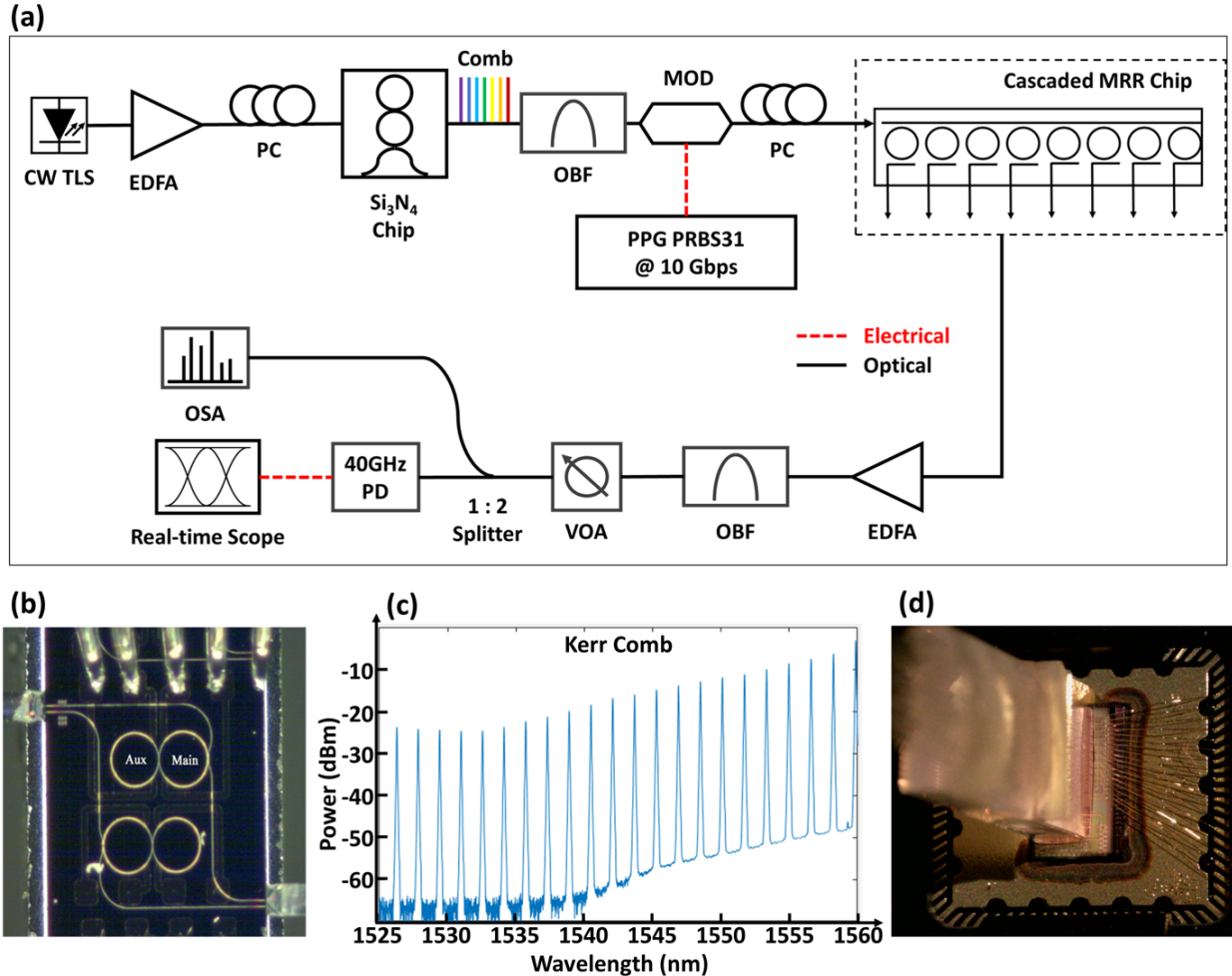


Figure 5.7: (a) Experimental setup for the comb laser. (b) Micrograph of Kerr comb source. (c) Output of Kerr Comb source. (d) Micrograph of packaged 1×8 microring resonator switch.

In our experimental testbed, we demonstrate modulated Kerr frequency comb lines and multi-wavelength optical switching achieved by wavelength-selective SiP MRRs. Fig. 5.7(a) illustrates our experimental setup: 1) a continuous-wavelength laser centered on 1561.42 nm is used to pump a silicon-nitride Kerr comb operating in the normal group velocity dispersion (GVD) regime [70]. The comb then generates evenly spaced lines at 200 GHz intervals, seen in Fig. 5.7(b) and (c), with a spectral flatness suitable for data communication applications [136]. 2) The output of the comb is filtered by an optical bandpass filter (OBF) to include only 22 channels. The output of the comb is then modulated with a 10 Gb/s PRBS31 via a linear reference modulator, and then coupled into our cascaded 1×8 MRR switch by means of a grating coupling fiber array (Fig. 5.7(d)). With each MRR exhibiting a free-spectral range (FSR) of 14.41 nm we can reliably drop multiple channels with the same element, given that the channels are spaced at ten-line intervals. 3) The dropped signals are coupled out of the photonics switch through the same fiber array, and then the signals are amplified through an in-line Erbium Doped Fiber Amplifier (EDFA) to compensate for coupling and equipment insertion losses. An OBF is then used to focus the signal into our wavelength range of interest, from where we can derive the optical spectra. Eye-diagrams are then captured from our 40 GHz photodetector-assisted real-time scope. This testbed demonstrates the feasibility of MRR to compose the multi-wavelength selective crossbar MRR switch proposed for our SiPAC design.

For this experiment, we focused specifically on wavelength-selective switching of broadcast signals as discussed in the previous sections. To demonstrate the efficacy of dropping multi-wavelength signals using a MRR, the first element (Ring 1) in our cascade microring switch is thermo-optically tuned to select the comb line at 1534.07 nm. Due to its 14.41 nm FSR, this automatically also selects the channel at 1548.48 nm. In Fig. 5.8(a), the optical spectrum captured at the drop port of Ring 1, we can see that our channels of interest dominate over all other lines, with a crosstalk suppression of 13.3 dB between our selected channel and an adjacent unselected one. Very similar performance can be observed for the next element in the switch (Ring 2), which is tuned to select wavelengths 1532.48 nm and 1546.87 nm; lines directly adjacent to the ones se-

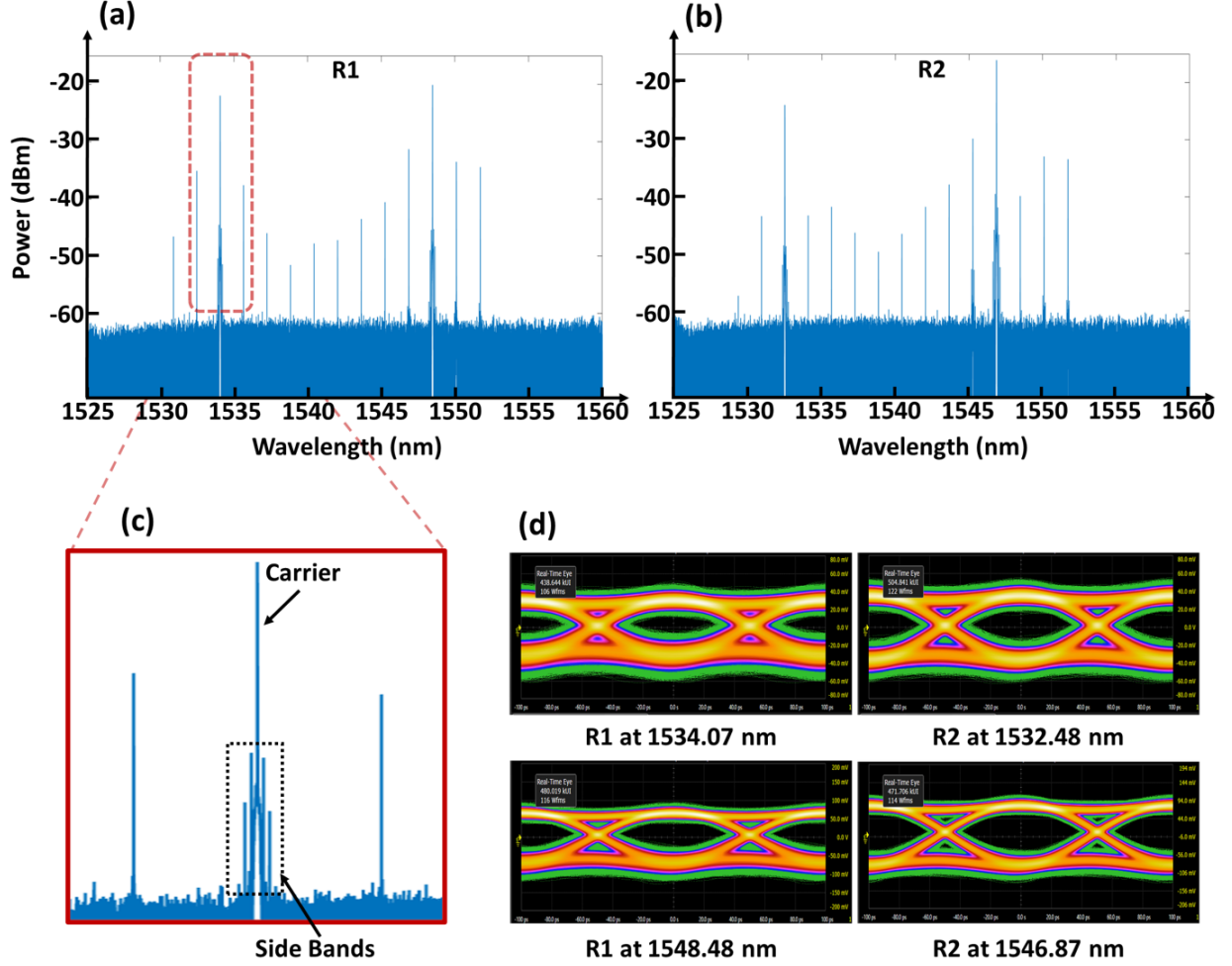


Figure 5.8: (a) Output spectrum of Ring 1. (b) Output spectrum of Ring 2. (c) Inset of (a), illustrating the sidebands surrounding the modulated carrier. (d) Output eye-diagrams of dropped signals.

lected by Ring 1. In its optical spectrum in Fig. 5.8(b), its minimum level of crosstalk suppression is observed to be 13.2 dB. Fig. 5.8(c) shows the modulated carrier and the surrounding sidebands.

By tightening the span of the OBF, we are able to capture the eye diagrams for each individually dropped line, without the interference from adjacent channels (Fig. 5.8(d)). We observe opened eyes in all cases, although a variance in the signal-to-noise ratios (SNRs) is observed: 6.02 dB to 7.23 dB, from R1 at 1534.07 nm to R2 at 1559.90 nm, respectively. This can be attributed to the uneven power of the comb lines. This factor degrades the relative extinction ratios of selected lines in relation to the amplified spontaneous emission (ASE) from the in-line EDFA, which establishes a

consistent noise floor across the spectrum. The collective results show the feasibility of using comb source and crossbar MRR switches to achieve the extreme high bandwidth optical broadcasting in our design.

5.2.5 System-scale Simulation

We use Netbench, an event-driven, packet-level simulator [119] to evaluate the performance of various architecture designs. We extended Netbench to support (1) topologies with varying link latencies and bandwidths and (2) traffic patterns with sequential job starting times that are found in collective communications.

DGX-SuperPOD [125]: The basic units of DGX-SuperPOD are DGX-A100 stations where 8 A100 GPUs are connected to an array of 6 NVSwitches using NVLinks [137]. Multiple DGX-A100 servers are then interconnected through a two layer leaf-spine fat-tree network using 8 InfiniBand host channel adapters (200 Gb/s) per server [125]. We therefore fix the inter-node bandwidth at $8 \times 200 \text{ Gb/s} = 1.6 \text{ Tb/s}$. We assume a $9\mu\text{s}$ NVLink latency [138] and 120ns InfiniBand switch latency [139]. We characterize the per-CU bandwidth here to be the sum of all intra-server (i.e. NVLink) bandwidths coming out of a single CU, similar to [132].

TPU 2D-Torus [126]: Google’s Cloud TPU system directly interconnects TPUs in a 2D toroidal mesh network [126] with uniform link bandwidth and latency. For systems with sizes that are not integer squares, we pick integer sizes for each dimension with minimum differences to achieve the targeted topology size. The per-CU bandwidth here is characterized as the total bandwidth a single CU has with its four neighbors.

SiPAC and BCube [134]: Since the SiPAC architecture is based on the BCube topology, we evaluate BCube built with EPSes in addition to SiPAC. While we choose r and L to best fit the required system size, we limit $r < 32$ and $L \leq 3$ to achieve realistic WSS radix and similar number of per-CU optical interfaces as the other topologies. The per-CU bandwidth for both architectures is characterized as the total bandwidth a single CU has with all connected switches (EPS for BCube and WSS for SiPAC) in each of the L layers.

We use all-reduce as a representative workload for both DP and MP to evaluate the performance of all architectures. We experiment with ring-based (R), mesh-based (M) and hierarchical ring-based (H) all-reduce algorithms on all the architectures. For hierarchical-based all-reduce, we set k to be equal to the number of nodes in a physical group in the topology (i.e. 8 in SuperPOD and \sqrt{p} in 2D-Torus). For the SiPAC architecture and BCube architecture, we also use the SiPAC all-reduce algorithm.

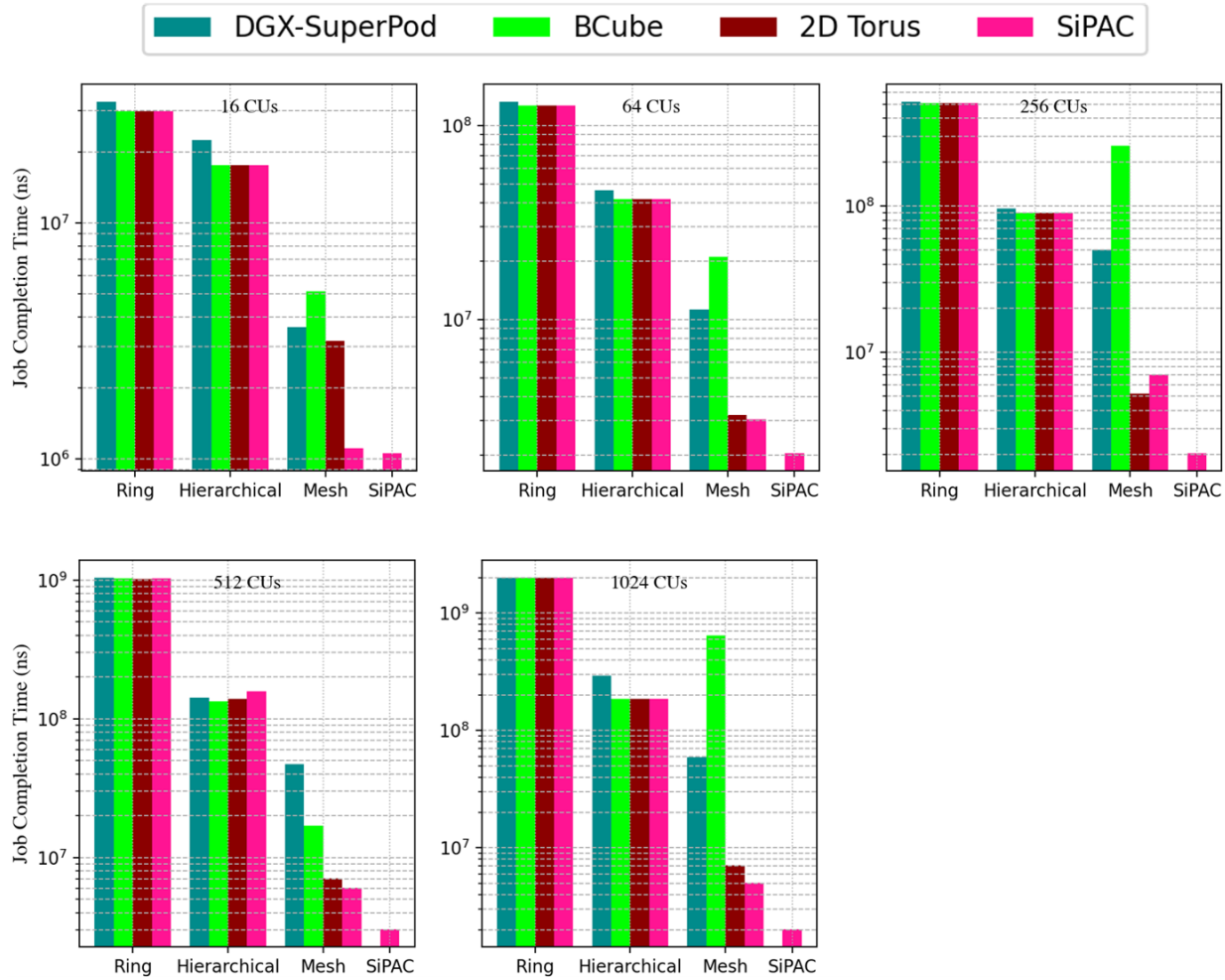


Figure 5.9: Job completion time of different topology-collective combinations for varying network sizes. The message size is set to be 1MB.

We evaluate the performance of all architectures using job completion time (JCT): the amount of time it takes for a specific collective communication job to finish. A lower job completion

time indicates better network performance. We use ECMP routing on all the topologies. We do not model any computation time since the main goal of this evaluation is to show improvement in training efficiency by minimizing the communication cost. We assume that any improvement in communication time could help enhance the overall training efficiency when computation and communication cannot be efficiently overlapped for large scale training [129].

Figure 5.9 demonstrates the performance of different topology-collective combinations at varying topology sizes. Here, the message size is set to be 1MB. The ring all-reduce performs the worst on each topology since it incurs the highest latency cost. We observe that for both ring all-reduce and hierarchical all-reduce, the JCT are similar for different topologies. This is because these two collectives only allow send and receive from one other node at each time step, which leaves many links under-utilized for these topologies with multiple connections per CU. This is not the case for mesh all-reduce and SiPAC all-reduce since these two collectives can take advantage of the multi-port property of the CUs in these topologies. While the JCT for ring all-reduce increases by over two orders of magnitude, the JCT for the SiPAC all-reduce remains relatively constant as the topology size increases. This is due to the linear dependency of the latency term on the topology size for the ring algorithm which dominates over the bandwidth term at large topology sizes. The hierarchical and mesh-allreduce both trade higher bandwidth cost with lower latency cost, which allows them to do well at lower message sizes.

5.2.6 Conclusion

In this work, we propose the SiPAC architecture which co-designs a silicon photonic enabled optical network topology with a specialized collective algorithm to optimize the communication latency involved in DDL communication. Using silicon photonic multi-wavelength selective switches, we introduce a hierarchical all-to-all topology while maintaining high bandwidth among all node pairs connected to the same switch, allowing for fast collective communications. The proposed WSS switch design is demonstrated on a physical testbed that showed its capability in achieving compact and high bandwidth multi-wavelength switching, with the SNR of switched sig-

nals attaining 7.23 dB. We compare our proposed architecture with representative DL acceleration topologies. The SiPAC architecture shows better performance at system scale.

5.3 Chapter Summary

This chapter proposes two disaggregated architectures. The first architecture benefits distributed deep learning training models with moderate size. It aims to address the system fragmentation issue when many of these training workloads are being distributed across a GPU cluster. The introduced optical switching layer between the compute resources and the aggregator switches can defragment the distant nodes with heavy communications. By doing so, higher layer traffic that goes through low bandwidth Ethernet or InfiniBand networks can be avoided. For the second architecture, training extreme large model is being considered under this scenario. The communication between hundreds or thousands of nodes has to go through the higher layer network. The proposed architecture then suggests to leverage extreme high bandwidth transceiver and switch technology to improve the communication efficiency between distant compute units. With the new interconnect technology and proper communication algorithms, the training speed of an extreme large deep learning model across can be improved.

5.3.1 Future Work

Section. 5.1 only shows that the proposed architecture can work at small scale. For system-level evaluation, methodologies and approaches shown in Chapter 4 can be applied to this work as well. The main difference is that for this disaggregated architecture, there is a bandwidth mismatch between Ethernet/Infiniband network layer and the GPU's NVlink. This is not captured in the current testbed. Besides, a multi-channel link needs to be established to show the path towards high aggregated bandwidth per link. FPGA-emulated accelerators can be considered as well. FPGAs are more flexible to show multi-channel link capabilities and capable of generating traffics based on more realistic datacenter traces. Furthermore, bulk 3D-MEMs would not fit into tightly-coupled datacenter integration. SiP MEMS switches should be considered for future system

demonstrations.

Section. 5.2 shows a complete architecture and algorithm co-design to address the network bandwidth issue in today’s GPU clusters. As described in the previous section, MRR based cross-bar wavelength-selective switch is used to achieve an optical broadcasting functionality for the SiPAC architecture design and for the corresponding hierarchical reduce operations. However, the design is limited to a fix bandwidth between each CU. Once the radix of the switch and the number of comb lines are decided, the CU-to-CU bandwidth is determined as well. Effectively, every CU-to-CU bandwidth in the SiPAC topology is fixed. DL training in a SiPAC cluster has to take this into consideration. A more flexible interconnect design that can steer one CU’s bandwidth and distribute more or less bandwidth to another CU can open up more design explorations. A Flex-SiPAC architecture can not only be the best fit for the hierarchical training strategy, but also be suitable for other training strategies. For example, if a Flex-SiPAC architecture can steer more CU’s bandwidth to their neighbor CUs, the architecture will certainly be more efficient for running ring all-reduce based algorithms than the static SiPAC.

Chapter 6: Fast Lanes for Expedited Execution at 10 Terabits

6.1 Introduction

Today's network interface cards are operating at $100 - 1000 \times$ slower data rates compared to other components, such as memory and GPU shown in Fig. 1.4. This gross mismatch significantly slows down applications can run across many nodes and have strong communications between them. The Fast Lanes for Expedited Execution at 10 Terabits (FLEET) is an initial platform designed for addressing the interconnection bottleneck in computing and network subsystem for the FastNICs program [140]. FLEET includes architecture, hardware, and software innovations to meet or exceed the FastNICs program's goals. We introduce FLEET cluster architecture and show development progress on FLEET's key hardware innovations - Optical Network Interface Cards (O-NICs). Detailed software overview can be found in Ref. [141].

Figure 6.1(1) shows interconnections between FLEET clusters. The aggregated interconnection bandwidth is 3 Tbps for generation 1 (Gen 1) system and 12 Tbps throughout the generation 2 (Gen 2) system. The clusters are connected through optical links using MEMS OCSs. As shown in Fig. 6.1(2), each cluster includes servers and PCIe devices (GPUs and NVMe storage drives) equipped with the O-NICs for communications between any two servers, a server and a PCIe device, or two PCIe devices. Each GPU or NVMe RAID controller owns a O-NIC and each 8 CPUs are with 3 O-NICs, as shown in Fig 6.1(3) and (4) separately. Figure 6.1(5) breaks down the connection of 3 O-NICs to a server node. The O-NICs are plugged into the PCIe slots to extend PCIe communication channels into the optical domain. A single O-NIC uses 16 wavelengths to support a 16-lane PCIe communication channels with two optical ports. This allows steaming input from one device while sending full bandwidth output to another device. Instead, communications will be limited to one peer with only one optical port each O-NIC. In addition, the use of FPGA allows

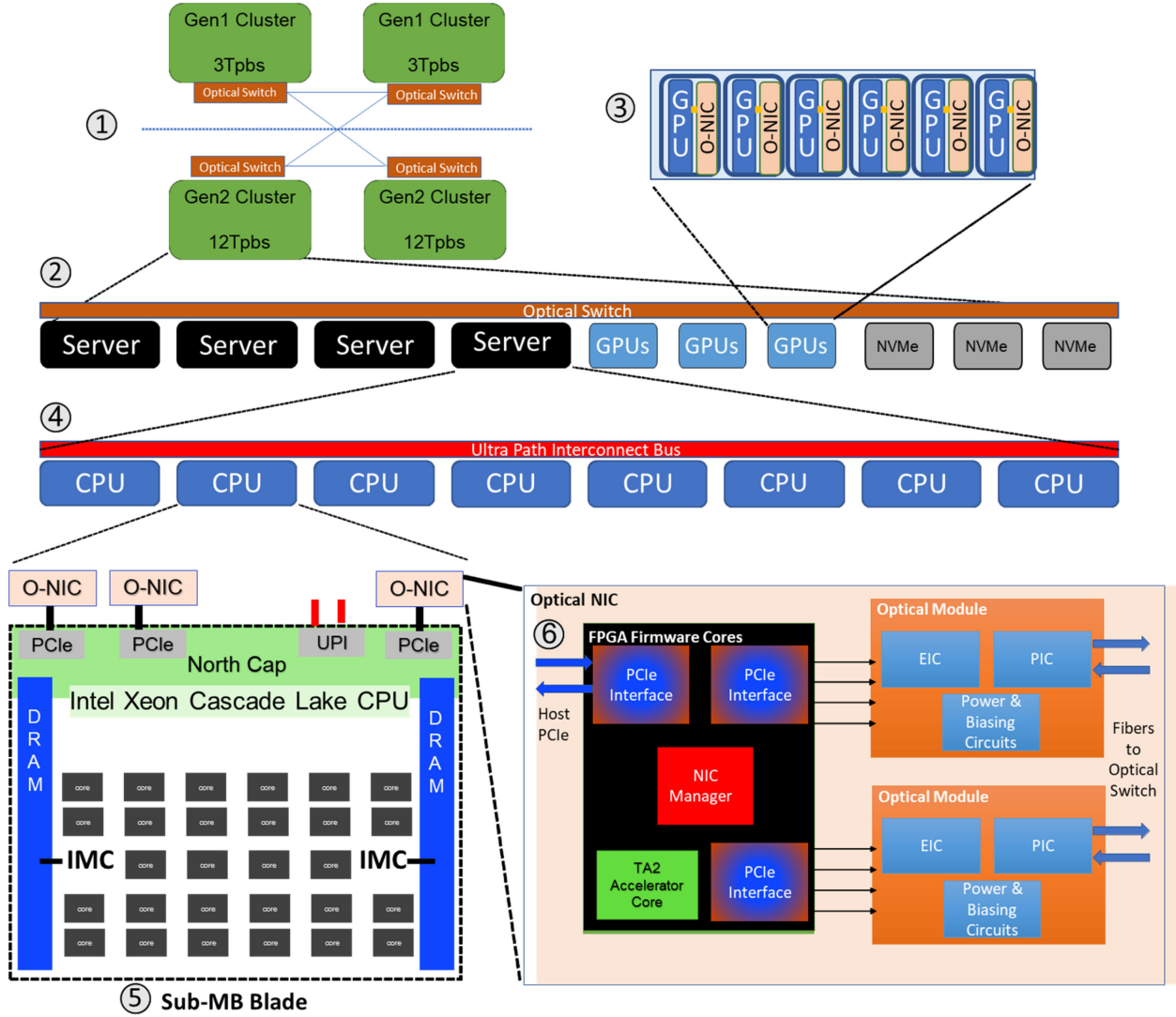


Figure 6.1: FLEET hardware overview, reprinted from [141].

in-place processing for data computation to accelerate certain applications and for minimizing stage-to-stage communication latency. Figure 6.1(6) illustrates the O-NIC.

6.2 ONIC Module Overview

6.2.1 Overall Packaging Plan

The O-NIC module is under development, and we report the current progress on each component of the entire package. Figure 6.2 illustrates the overall O-NIC packaging plan. The target

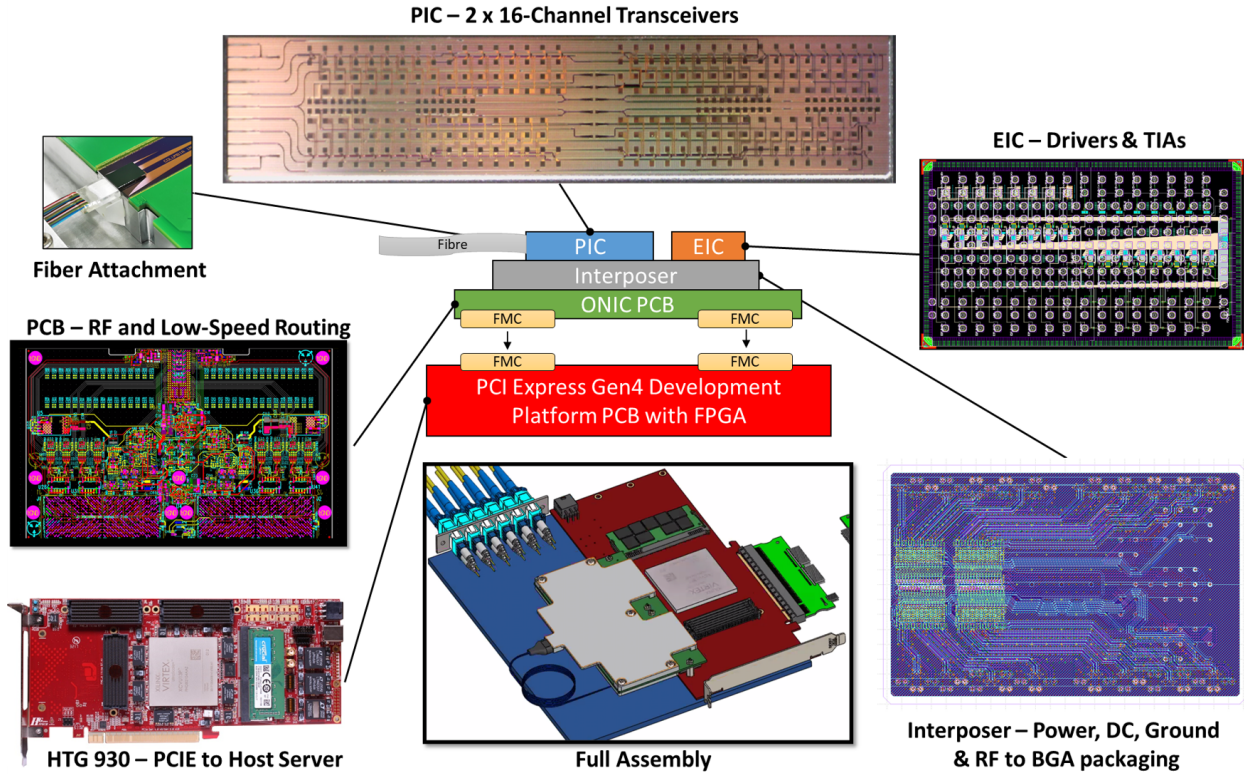


Figure 6.2: FLEET O-NIC packaging plan.

package is leveraging 2.5D integration. The transceiver PIC chip contains two 16-channel transmitter and receiver links to match the 16-lane PCIe communication link as described in Section 6.1. Each driver/TIA EIC chip can modulate or receive 8 channels and in total four identical EIC chips are used to interface with the PIC chip. Both PIC and EIC chips are designed to be flip-chip bonded to a multi-layer ceramic interposer. The interposer's high speed transmission lines are properly modeled and structured for driving/receiving RF signals with both PIC and EIC. For RF signals between EICs and the O-NIC PCB and DC signals are fan-out through the interposer to its bottom side. The interposer then is mounted to the O-NIC PCB via BGA. Fiber attachment is necessary to couple light in to/out of the PIC chip. On the O-NIC PCB, high speed RF signals are directly routed to the FPGA's transceivers through FMCs. DAC/ADC circuits are used for the feedback control of PIC chip, and are configured by the control logics in FPGA as well. More details on the PCB are included in Section 6.4. The packaged O-NIC PCB will be plugged into

FMC connectors on an HTG-930 FPGA evaluation board which will be connected to the server through PCIe. LC connectors are envisioned for the connection to other O-NICs. The packing plan is compatible with other FPGA evaluation boards that have the same placement of the two being-used FMC connectors.

6.2.2 Link Analysis

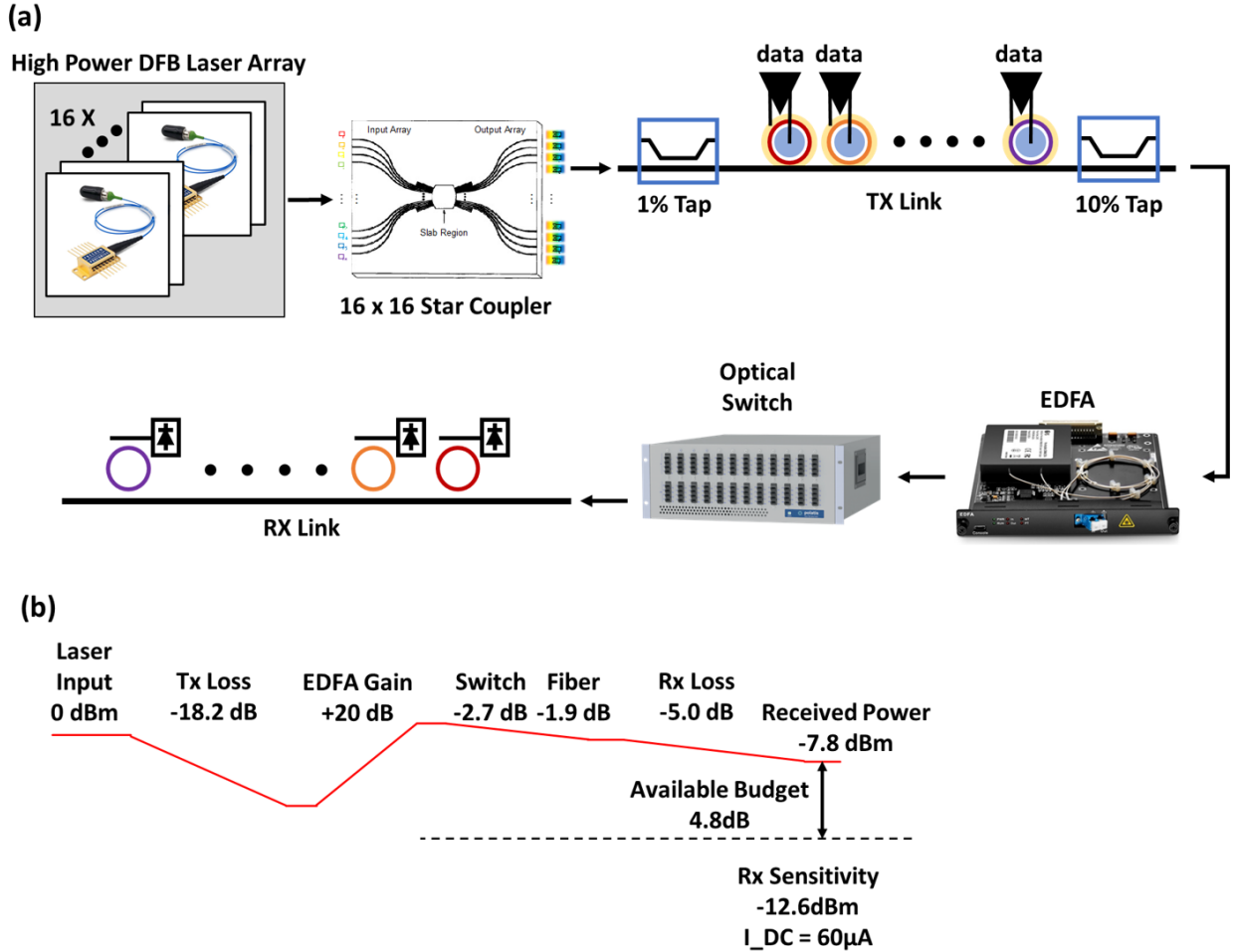


Figure 6.3: FLEET O-NIC link and power budget.

The design of the link is targeted at 256 Gb/s with 16 optical channels operating at 16 Gb/s. The spacing between channels was subject to variation and optimization based on the intermodulation crosstalk of modulators and crosstalk penalty of filters. The channel crosstalk penalty in the link

based on the PDK models was modeled and analyzed using Lumerical Interconnect. The minimum required spacing between channels is set to 150 GHz. The design was considered to have 16 Gb/s data readily available for each channel and for all 16 channels. Figure 6.3(a) shows the source, loss, and gain components in the link. The loss components include the modulation loss, taps for monitoring, coupling loss, optical switch's loss, fiber loss, and filter loss. To meet the receiver's sensitivity requirement, an EDFA is needed to compensate the loss. Assuming each optical channel's input power to the TX is 0 dBm. A summary of the optical budget analysis for this design is presented in Fig. 6.3(b). The sensitivity of receiver at 16 Gb/s was simulated and set to be -12.6 dBm. We consider the variations of the laser source, star coupler, and EDFA gain. A 4.8 dB margin has been achieved considering a 20 dB gain EDFA.

6.3 Silicon Photonic Transceiver Chip

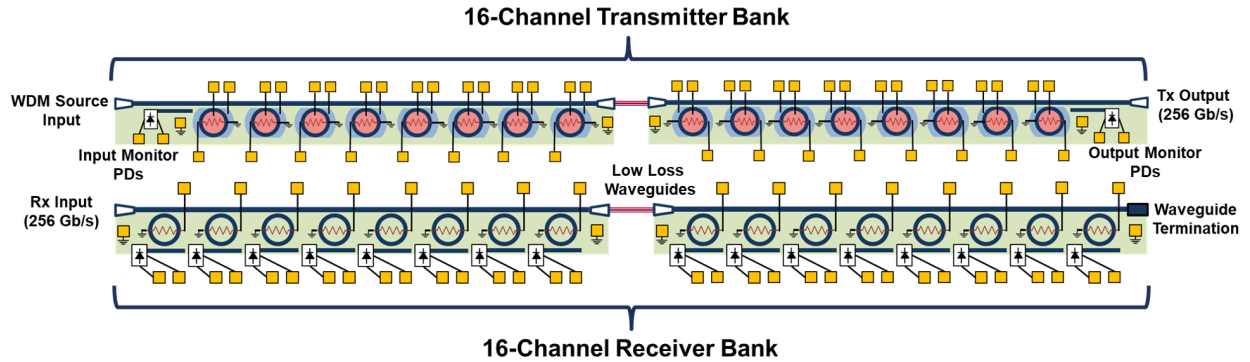
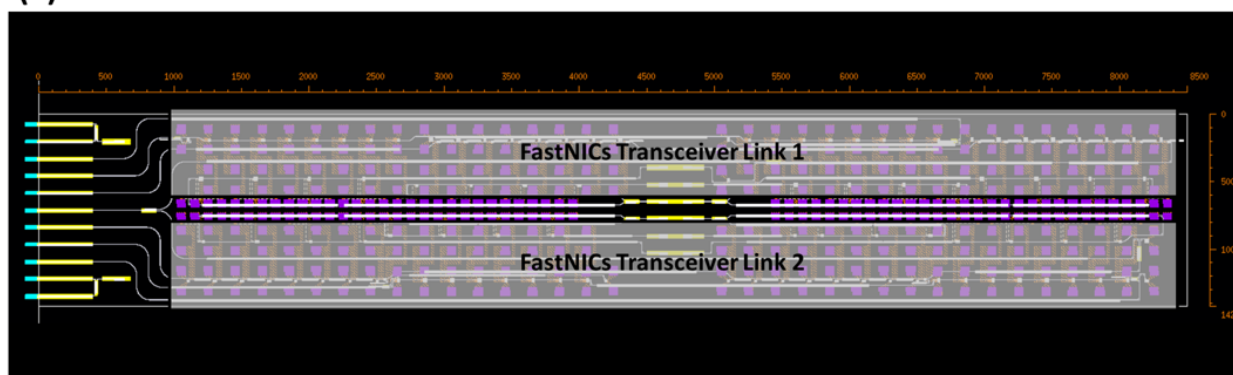


Figure 6.4: FLEET PIC transmitter and receiver banks.

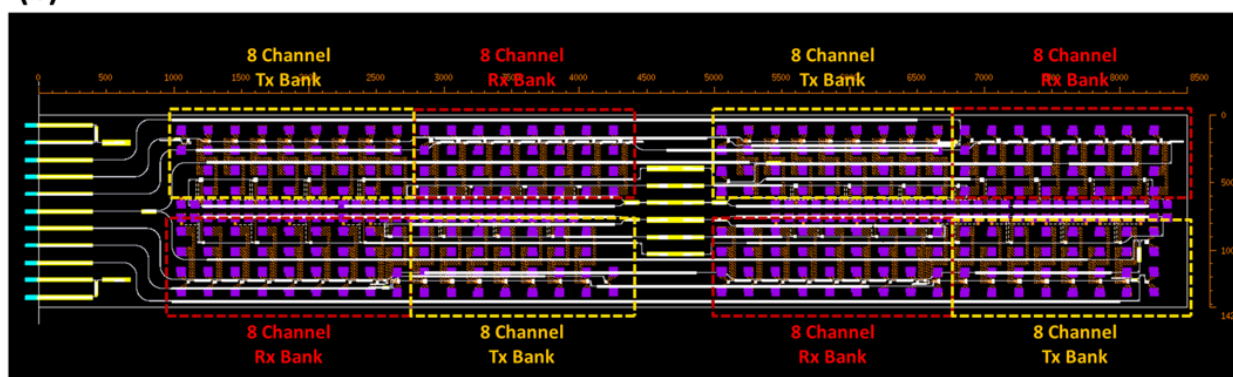
The PIC transceiver chip requires 256 Gb/s transmitter and receiver per link, operating at 16 Gb/s per wavelength. For the transmitter bank, a 16-wavelength WDM source will enter the chip as the input. Microdisk modulators are used to modulate the light at each wavelength. Before and after the microdisk modulators, input and output monitor PDs are included to enable thermal control and polarization control. For the receiver bank, microring filters are used to select each corresponding modulated signal and forward it to the receiving PD. Monitoring PDs are not needed

for the receiver bank, since the receiving PDs can also provide the DC power level information. Low loss waveguides are used in between every eight cascaded elements. Figure 6.4 depicts the 16-channel transmitter and receiver banks. Test structures are included to assess fabrication variations on different sections of each PIC.

(a)



(b)



(c)

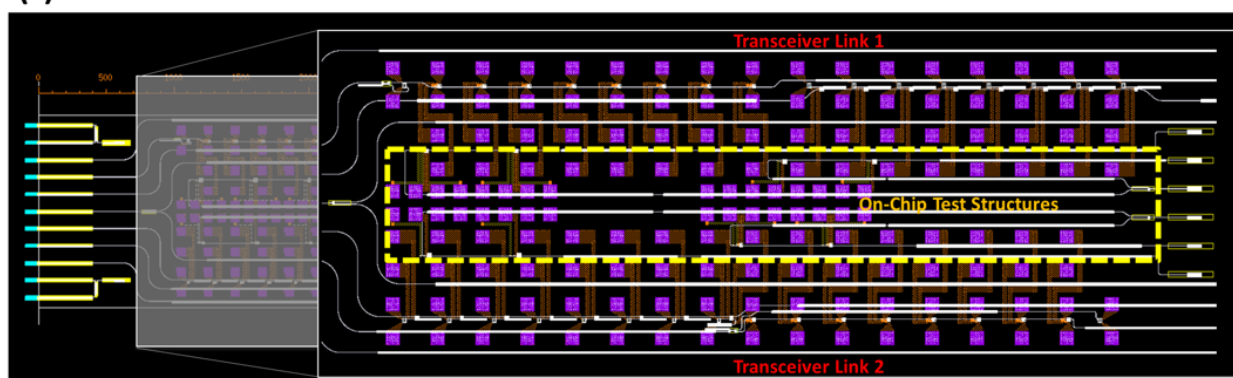
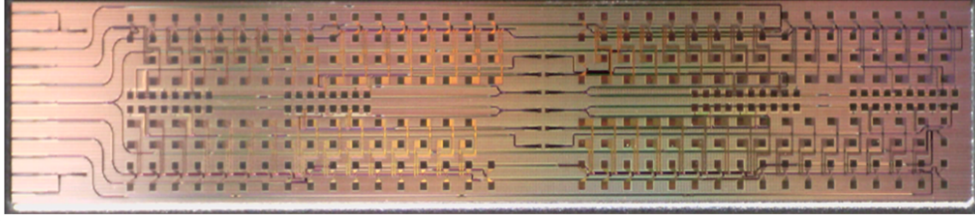


Figure 6.5: FLEET PIC floor plan. (a) The two 16-channel transceiver links. (b) The transmitter and receiver banks. (c) Two 8-channel transceiver banks and test structures.

(a)



(b)

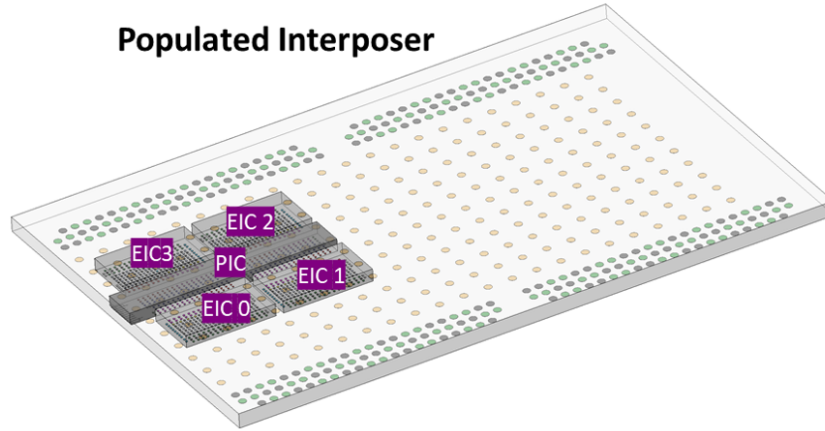


Figure 6.6: (a) FLEET PIC chip.(b) Illustration of a populated interposer.

As shown in Fig. 6.5(a), two transceiver links are placed on the PIC, and they are divided horizontally across the middle. On the left, edge couplers are used to couple light into and out of the PIC chip. Each transceiver link is then divided again into two 8-channel Tx/Rx banks as shown in Fig. 6.5(b). This ensures the compatibility with a modular 8-channel driver/TIA EIC design. The pad pitch is designed for acceptable crosstalk performance for high-speed signals while minimizing the entire PIC chip size. Between the transmitter/receiver banks on the same side, low loss waveguides are used to reduce the propagation loss within the chip. Figure 6.5(c) shows a magnified picture of the left half of the chip. On-chip photonic test structures are in the middle and they can be probed optically and electrically without damaging actual transceiver links. The test structures can show the thermal tuning range of optical components on each chip, and the intrinsic resonance of the microdisk resonators and the tunable microring filters. Based on the measurements of the test structures, we are able to select PIC chips for further packaging. The

design results in a photonic chip area of 12.11 mm^2 ($1.425 \text{ mm} \times 8.5 \text{ mm}$).

The fabricated PIC chip is shown in Fig. 6.6(a). Figure 6.6(b) illustrates how to populate the PIC chip and the EIC chips on an interposer.

6.4 Development of O-NIC PCB

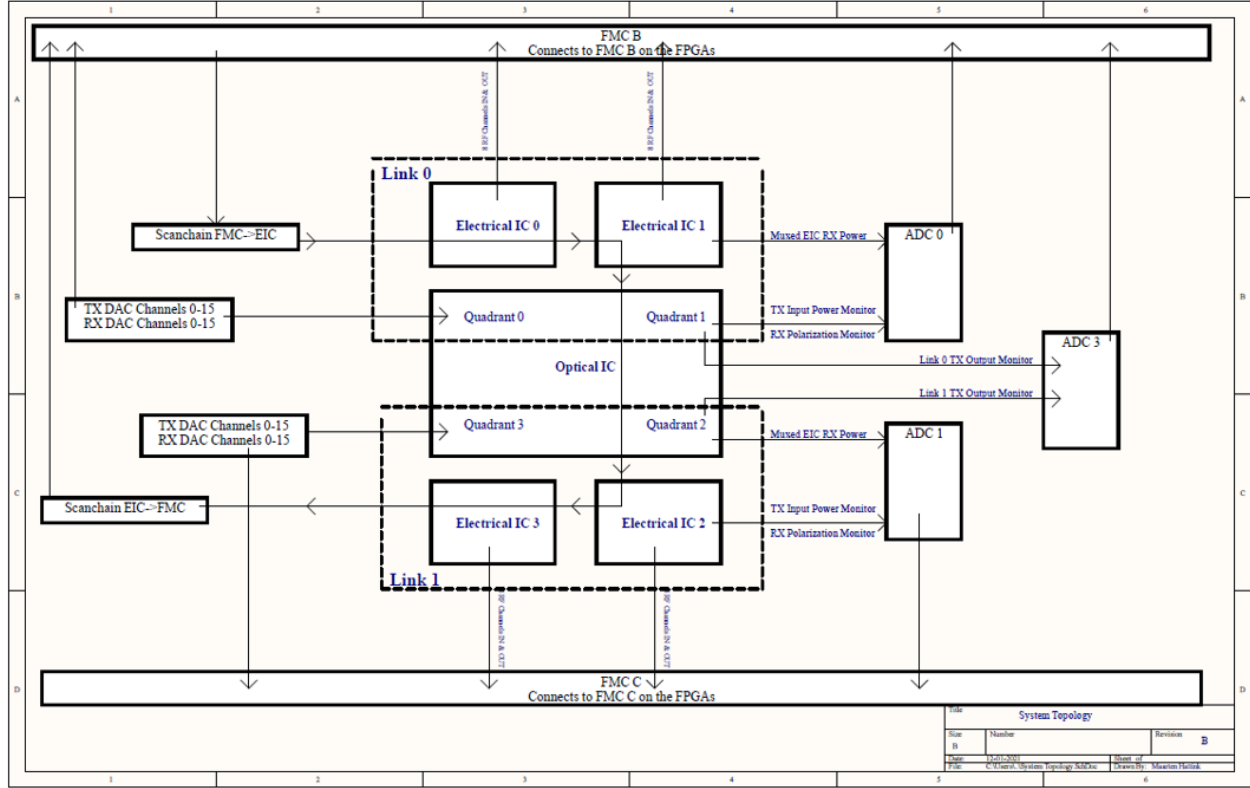


Figure 6.7: (a) FLEET PIC chip.(b) Illustration of a populated interposer.

The O-NIC PCB supports both high-speed RF differential signals, low-speed control signals, and power planes for the PIC and EIC chips. As shown in Fig. 6.7, the TX/RX RF differential signals for EIC #0 and #1 in Link #0 are routed to FMC B on the FPGA board, and the signals for EIC #2 and #3 in link #1 are connected to FMC C. AC coupling capacitors are needed to cancel the DC levels. The RF signals are then connected to the GTY transceivers in the FPGA. A scan-chain circuit is necessary to configure the EIC chips, while minimizing the wires by configuring in a sequential manner. The scan-chain circuit is connected to the FPGA through FMC B. For

biasing the heaters of both microdisk modulators and microring filters on the PIC, DAC circuits are designed and interfaced with FPGA through FMC B and C for link #0 and link #1 respectively. To close the feedback control loop, ADC circuits are designed for the TX input, TX output, and EIC receiver monitoring signals. TIA circuits are required for converting photo current into voltage before the amplification and ADC stages. The TX output feedback monitoring signals for both link #0 and link #1 are connected back to the FPGA through FMC B. For the rest feedback signals, they are routed through FMC B and C respectively. The current layout progress is shown in Fig. 6.8.

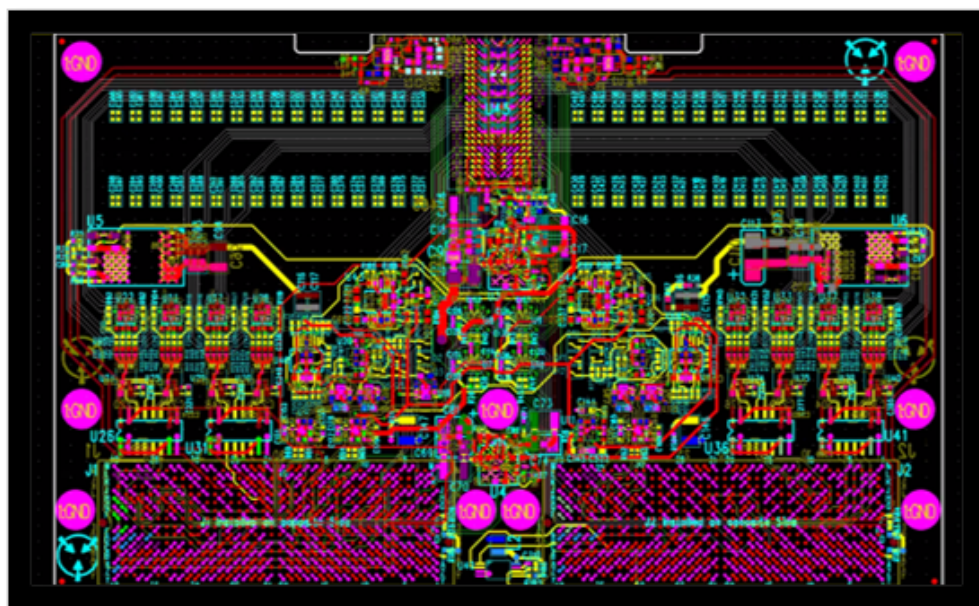


Figure 6.8: FLEET O-NIC PCB layout.

6.5 Test Packages

Test packages are necessary for the development of the FLEET O-NIC package. TX and RX test packages can be used to verify ADC/DAC circuits, implement the feedback control algorithms, test prototype EICs, as well as gain high-speed design experiences for the ultimate O-NIC PCB.

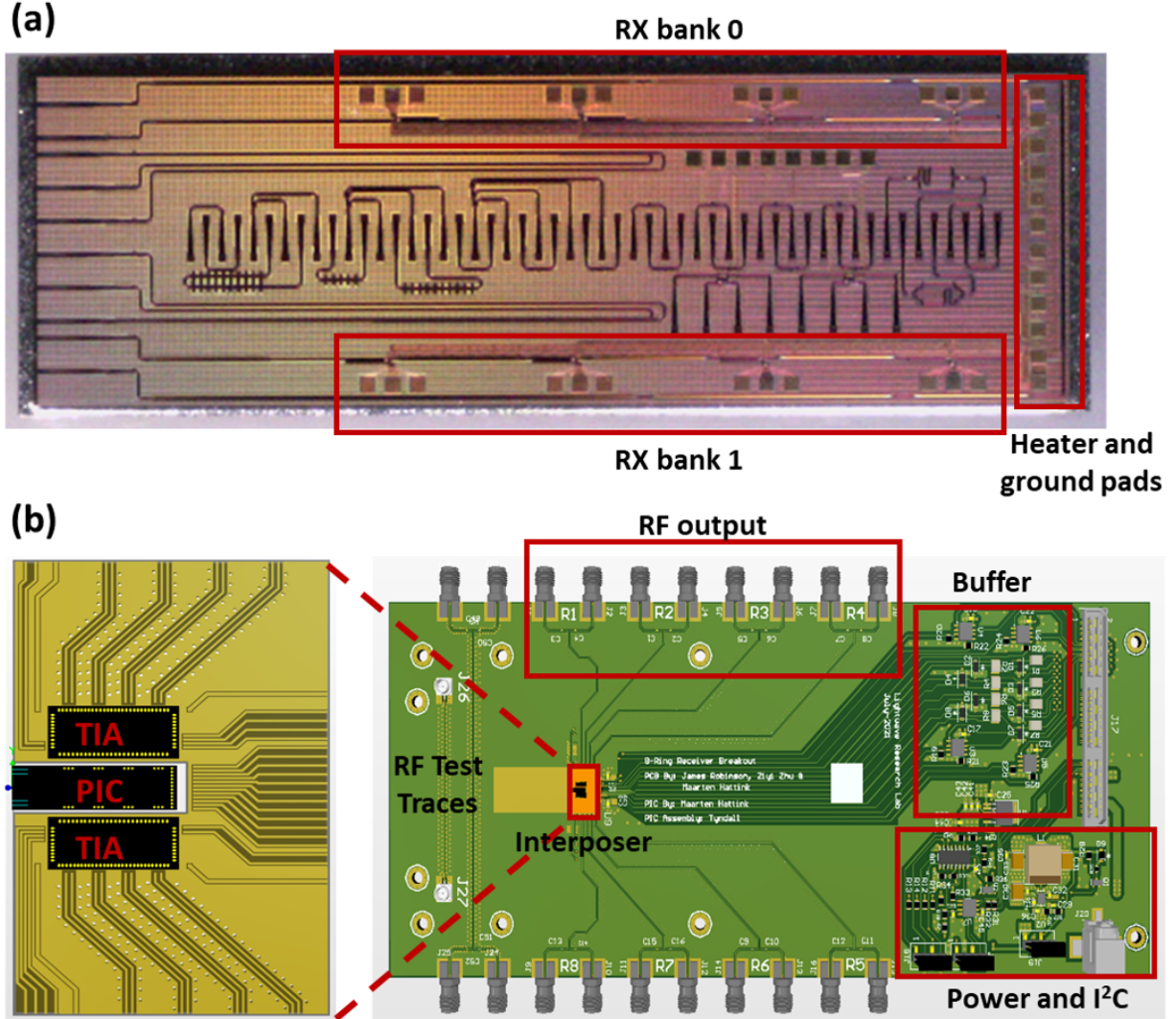


Figure 6.9: FLEET RX test PIC chip and packaging plan.

6.5.1 RX Test Package

The PIC receiver test architecture is built on a bus waveguide with microring filter elements. Eight microring filters are coupled to the bus waveguide. As shown in Fig. 6.9(a), four rings are located at the top and the other four are located at the bottom. The rings are separated by $750\text{ }\mu\text{m}$ apart to align with the corresponding channel on the TIA chips for packaging purposes. The heater and ground pads are located on the right side of the chip. At a high level, the packaging uses the 2D integration approach with wirebonds. An interposer, however, is also used to fan out the signals

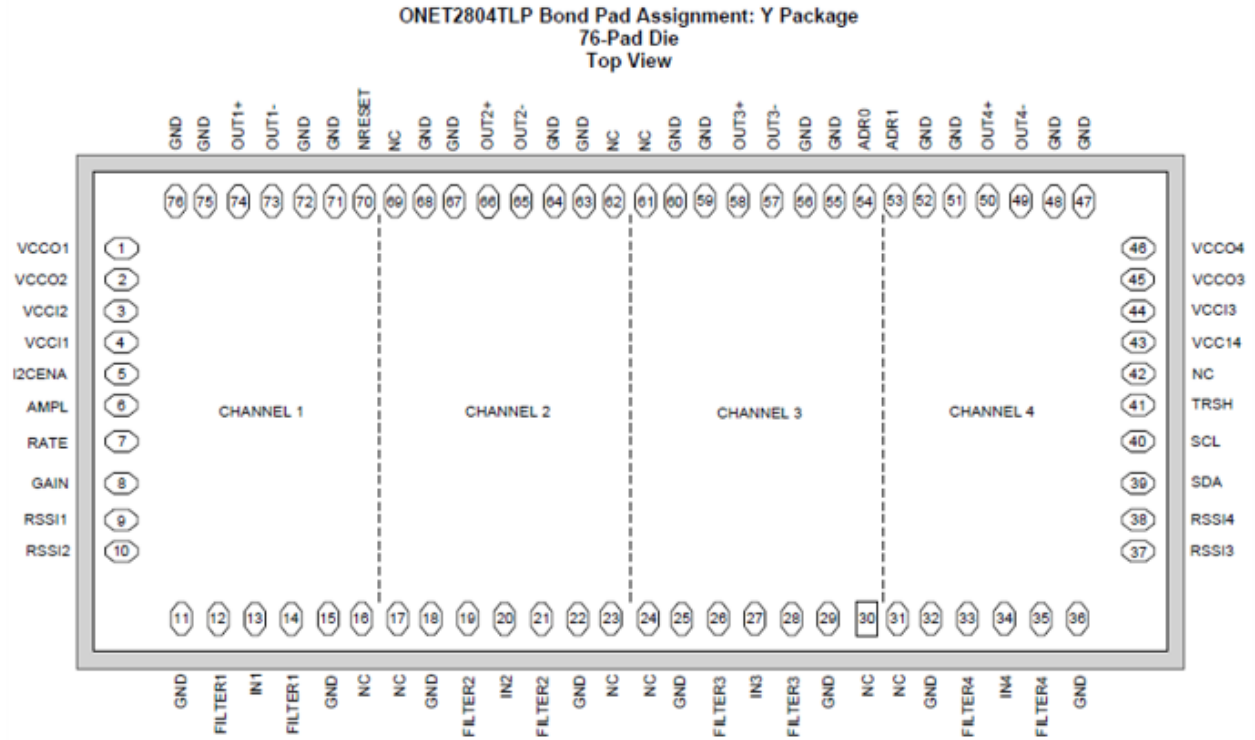


Figure 6.10: TIA bond pad assignment.

as well as provide the connectivity between the PIC/TIA chips and the PCB. The packaging plan is shown in Fig. 6.9(b). Two 4-channel TIA chips (ONET2804TLP) are placed on the interposer which is attached to the PCB underneath. There is a U shape cavity in the interposer to let the PIC to be attached to the PCB as well. The receiving RF photo current signals are wire-bonded from the PIC to the TIA. The TIA's RF voltage output is wire-bonded down to the interposer, and then gets fanned out to the edge of the interposer for another wire-bonding connection to the PCB. Finally, the RF signals are routed to the edge SMAs at both top and bottom of the PCB. Other buffer, power, I²C circuits are needed for the control of the TIA chips and the heater of the PIC chip. Figure 6.10 shows the bond pad assignment of the TIA chip. More details about the TIA chip regarding V_{cc}, featured Received Signal Strength Indicators (RSSIs), I²C interface, sensitivity, operating speed, and other parameters can be found in Ref. [142].

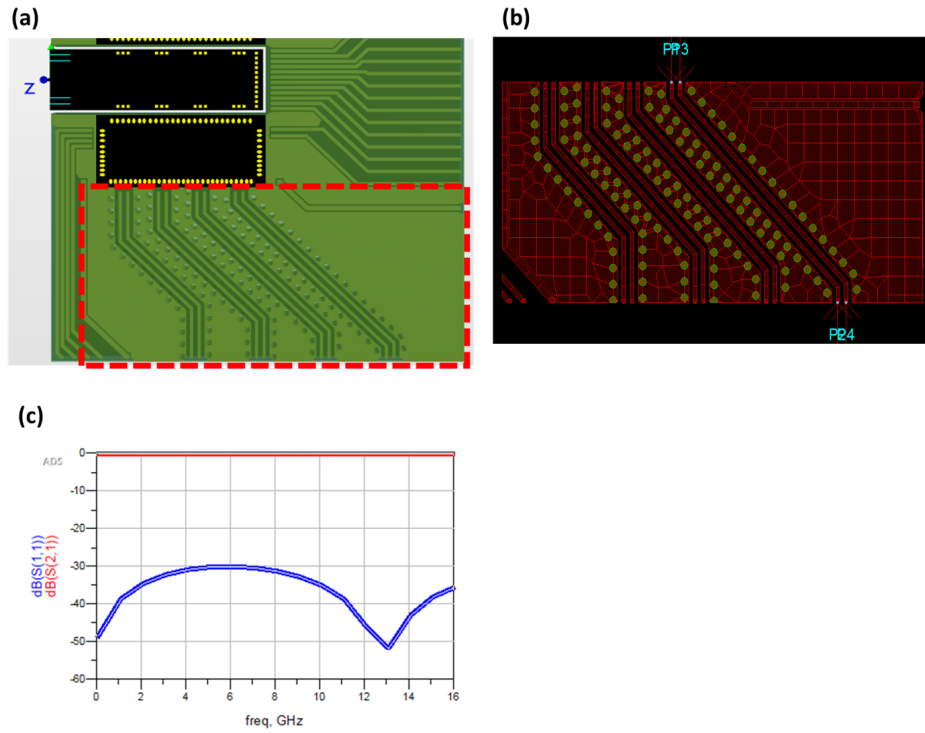


Figure 6.11: (a) Interposer differential traces. (b) ADS layout model. (c) S_{21} and S_{11} results.

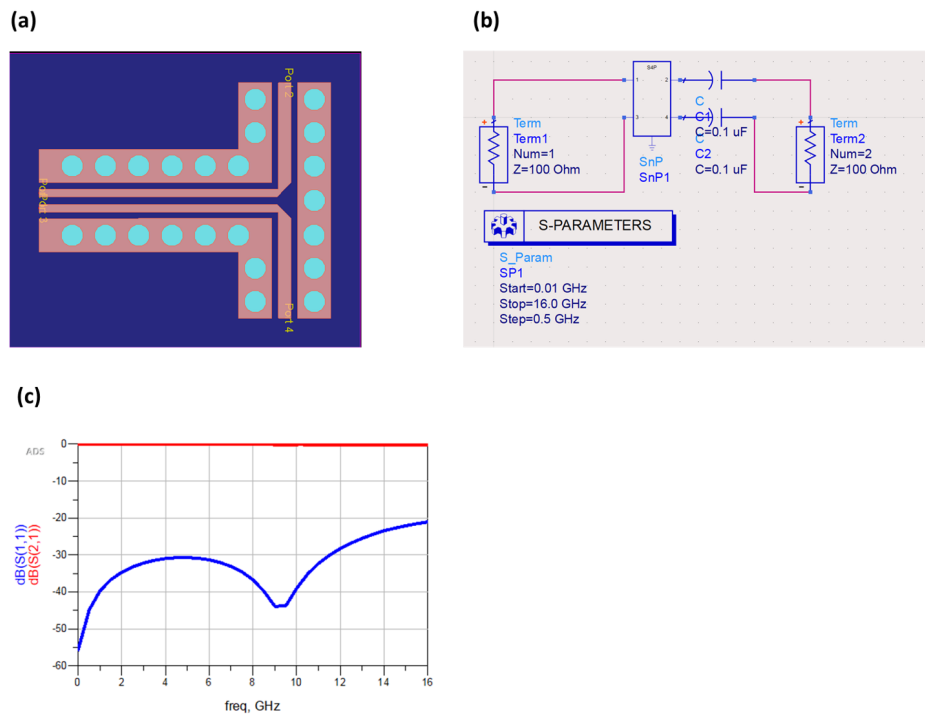


Figure 6.12: (a) ADS layout model for PCB differential to single ended traces. (b) ADS circuit model. (c) S_{21} and S_{11} results.

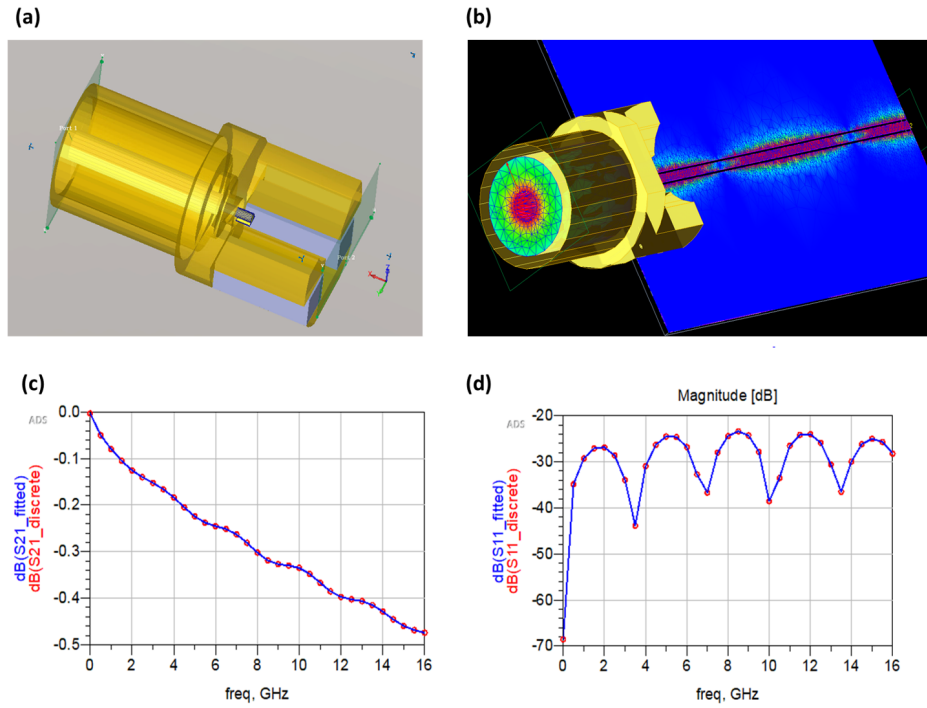


Figure 6.13: (a) ADS EMPro edge sma model. (b) 3D model with landing trace. (c) S_{21} result. (d) S_{11} result.

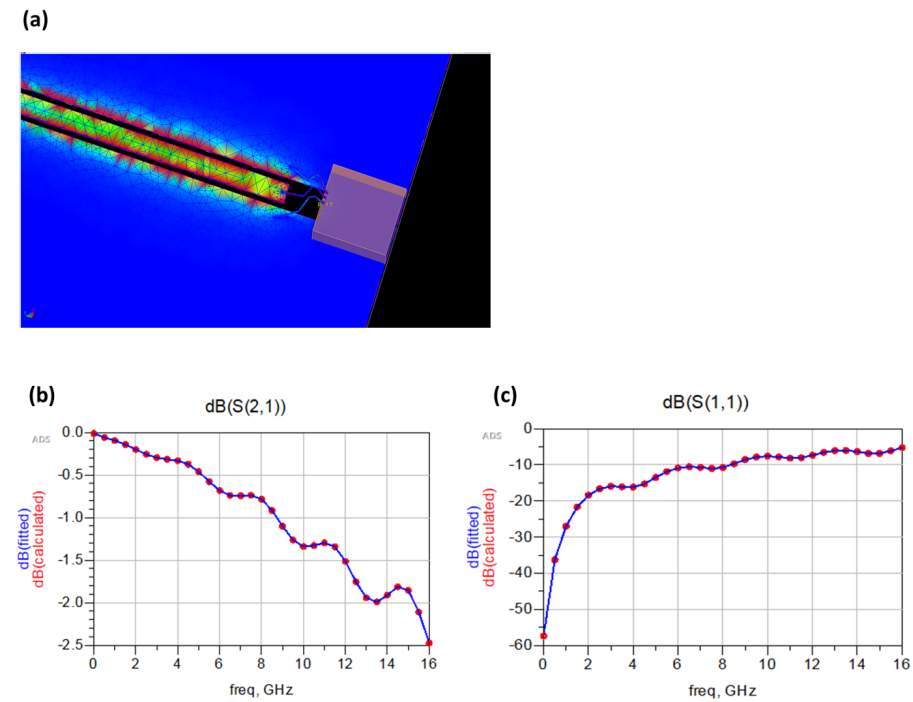


Figure 6.14: (a) ADS 3D wirebonds model. (b) S_{21} result. (c) S_{11} result.

To ensure error-free operations on the high-speed links, transmission lines, such as interposer traces, PCB traces, wire-bonds, and SMA connectors, need to be properly modeled and designed. The models are implemented in KEYSIGHT Advanced Design System (ADS) and simulated with 3D FEM simulations. We performed the simulations for the differential CPWG trace on the interposer, differential to single ended CPWG traces on the PCB, edge SMA, and wire-bonds. For the wire-bonds, we assume a 500 μm height and the horizontal distance between the pads is 700 μm . Figures 6.11, 6.12, 6.13, 6.14 show the models and the S-parameters results respectively.

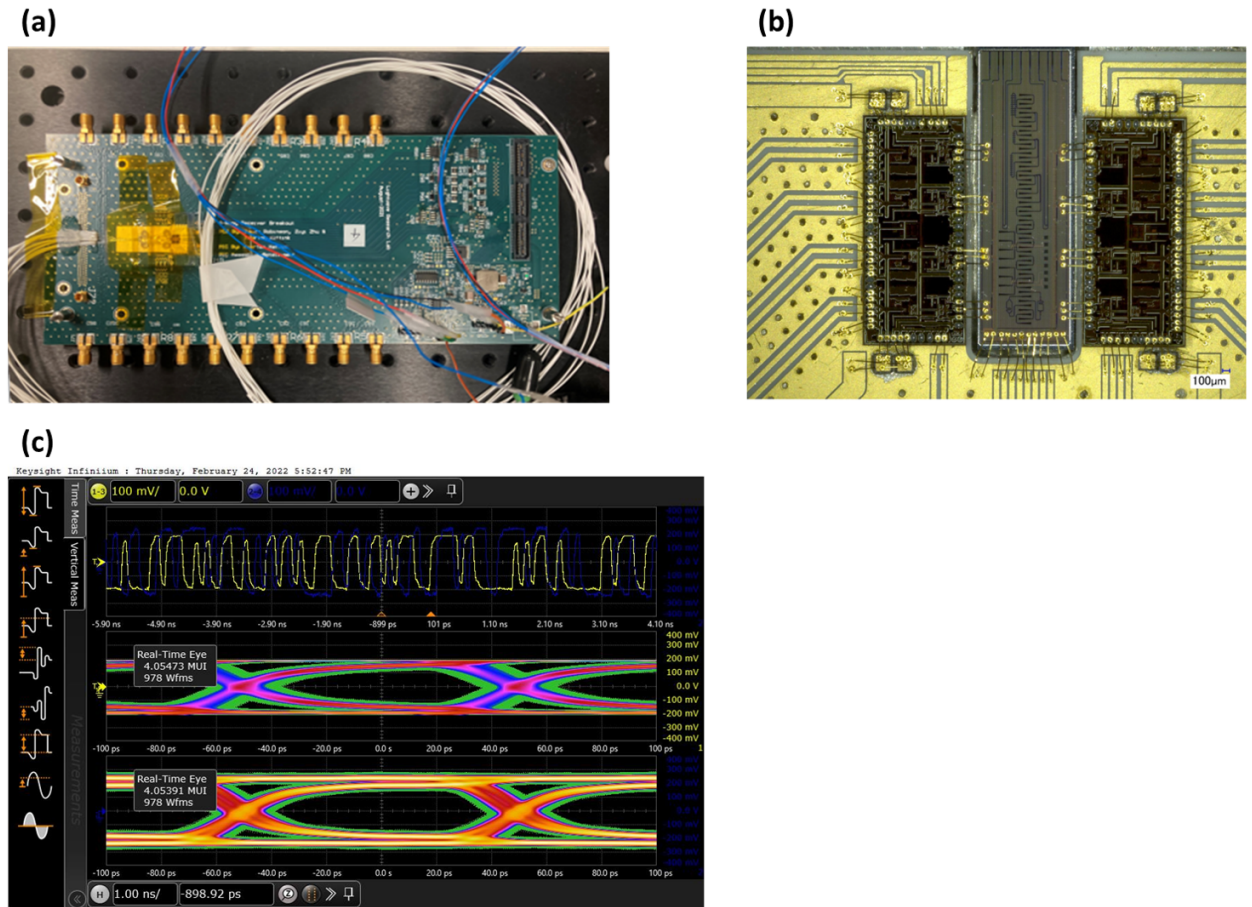


Figure 6.15: (a) RX test package full assembly. (b) Wire-bonding picture. (c) Eye diagrams.

The received full RX assembly is shown in Fig. 6.15(a), and Fig. 6.15(b) shows the wirebonds between PIC and TIA chips, and between TIA chips and the interposer. Two 10Gb/s PRBS 31 data streams are transmitted to the RX test package and the first and second microring filters are used to

drop the data signals. The corresponding edge SMAs are connected to a real-time scope through SMA cables. Open eye diagrams are observed as shown in Fig. 6.15(c). The streams are from two SFP+ transmitters driven by an FPGA.

6.5.2 TX Test Package

The PIC transmitter test architecture is built on a bus waveguide with microdisk modulator elements. The TX test package was developed through the same process as the RX test package. Critical RF components are simulated. Figure 6.16 shows the PIC transmitter chip, a packaged TX test board without driver EIC, and a packaging plan for a future version with a custom driving EIC. These follow the 2D integration approach without an interposer.

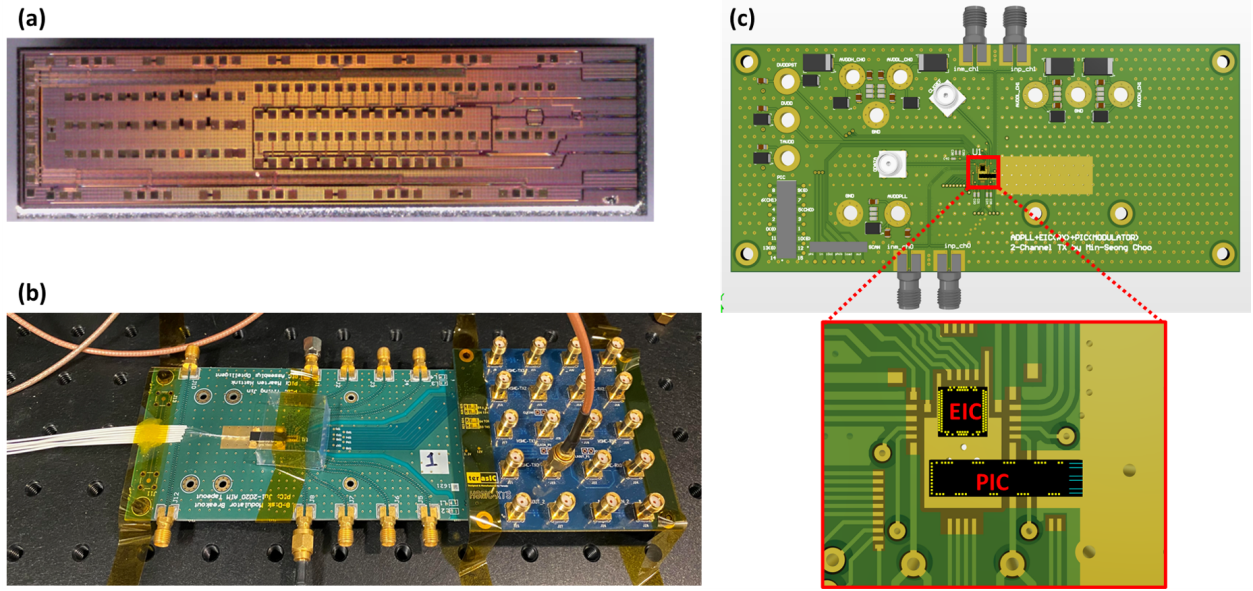


Figure 6.16: (a) TX test PIC chip. (b) TX test board. (c) Packaging plan with a driving EIC.

6.6 PCIe Interface

In order to transfer data from a server to the O-NICs or vice versa, an PCIe subsystem needs to be implemented on the FPGA for the communication. For the FLEET project, an PCIe subsystem with a streaming interface to the Aurora transceiver cores is required due to the latency requirement

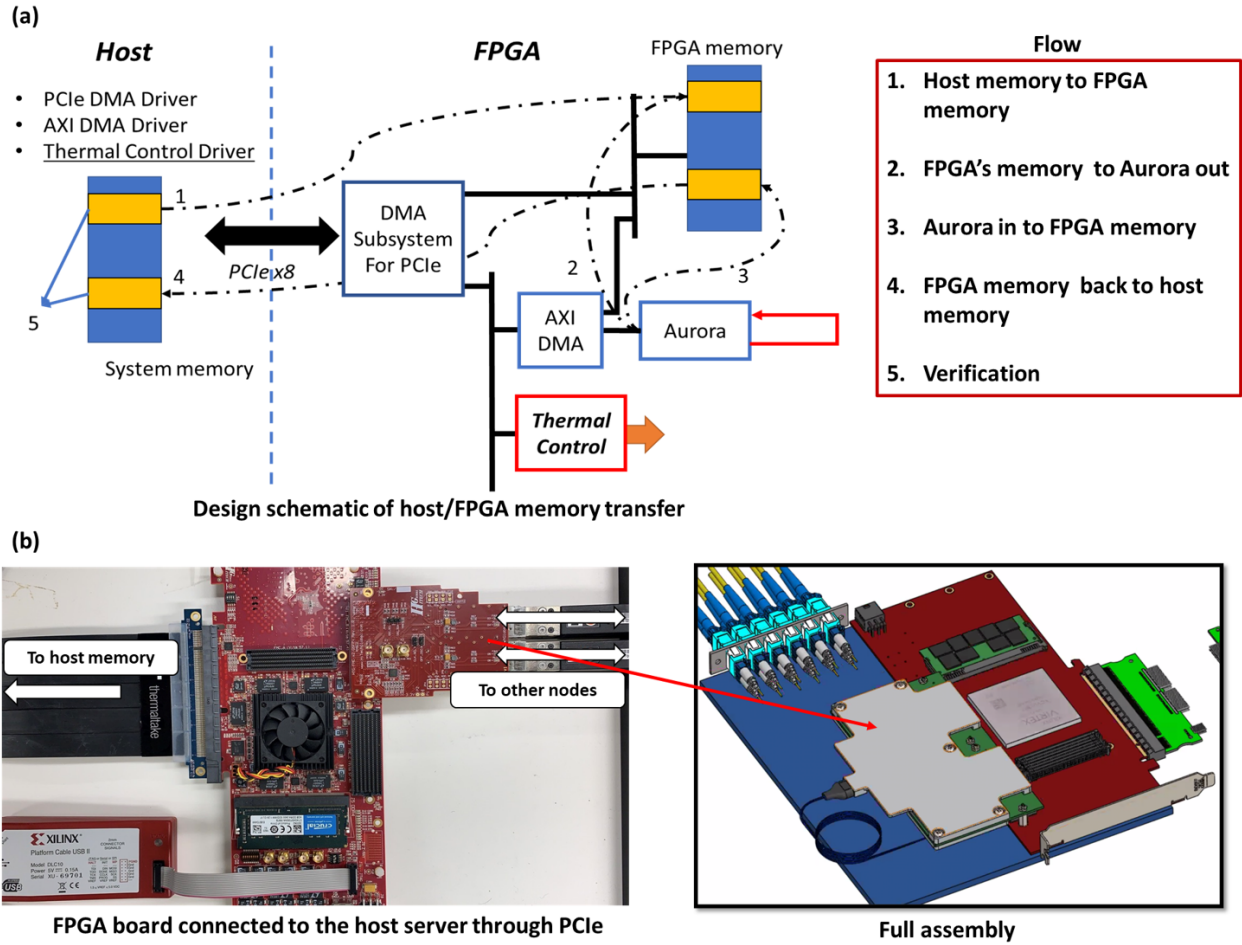


Figure 6.17: (a) Schematic of the system for the host and FPGA memory transfers. (b) Physical hardware.

of the project. We implemented a simpler version of the PCIe subsystem for system verification during the development process. Figure 6.17(a) shows the schematic of the system design, and the flow for memory transfer verification. A DMA subsystem for PCIe core is implemented to interface with the system through PCIe. The core also provides an AXI-4 memory mapped interface which is connected to the FPGA memory. Another AXI-lite interface from the DMA subsystem core is used to control an AXI DMA core and a thermal control core. The thermal control core will be implemented using Vivado-HLS and is under development. The AXI DMA core is used to fetch data from the FPGA memory and send out the data through the Aurora transceiver cores to the optical modules. The AXI DMA core is also responsible for receiving the data from the optical

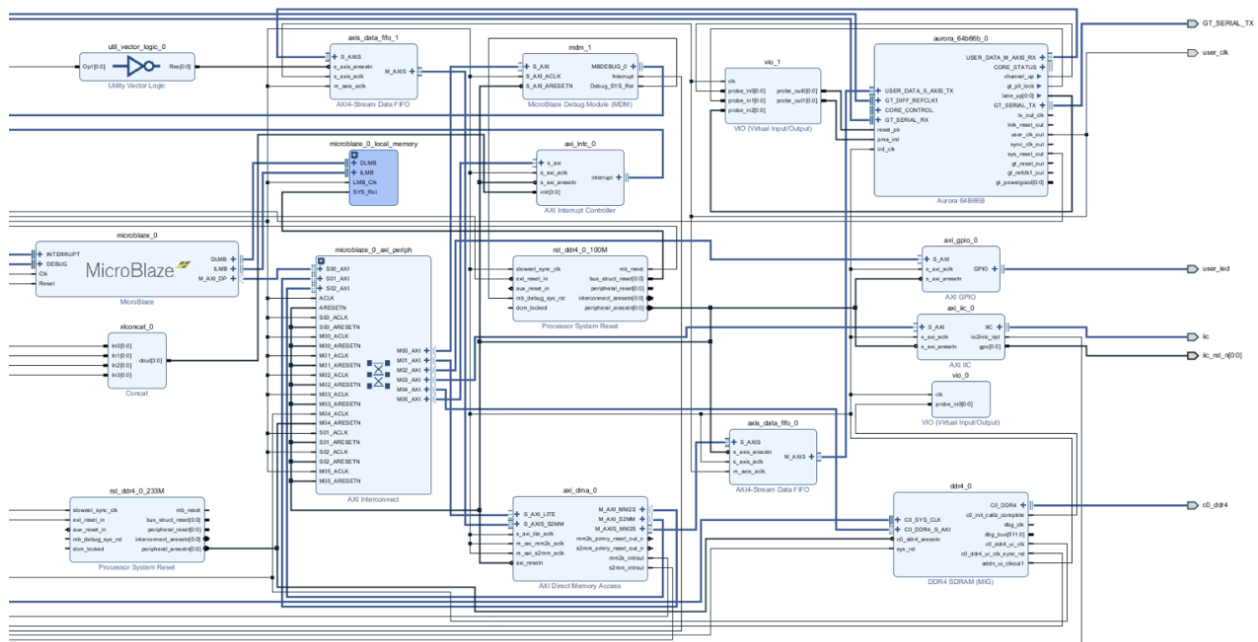


Figure 6.18: PCIe subsystem block diagram.

modules and store the data into FPGA memory. To achieve the memory transfers, PCIe DMA software driver and AXI software driver are implemented in the host. There are five steps for the memory transfer verification. First, move a chunk of data from the system memory to the FPGA memory through PCIe using the DMA subsystem core. Second, configure the AXI DMA to fetch the data and forward the data to the Aurora core and the transceiver modules. The transceiver modules are connected back-to-back in this case. Third, use the AXI DMA to receive the looped back data and store the data to another place in the FPGA memory. Fourth, configure the DMA subsystem core again and read the data back to another place of the system memory. Last, verify the transmitted and received data. This flow is also shown on Fig. 6.17(a) right. Figure 6.17(b) shows the physical hardware FPGA board, PCIe cable, and transceiver modules that are used in the implementation. We successfully demonstrated a working system for the data transfers. The transceiver modules will be replaced by the O-NIC modules in the future. Figure 6.18 shows the detailed block diagram of the system we described above in Vivado.

6.7 Chapter Summary

In this chapter, we introduces the FLEET architecture and the project goals. We the report the overall packaging plan of the hardware innovation - O-NIC module and demonstrate the fabricated SiP transceiver chip. The design of O-NIC PCB and the test packages are presented to show the path towards the full assembly. A PCIe interface is developed for future system functionality verification. The O-NIC module is still under development, and more details will be reported in future publications.

Conclusion

The work presented in this dissertation focused on developing various high performance silicon photonic interconnected system architectures for their deployment in future datacenter and HPC systems. We summarize the contributions of this dissertation, and then discuss future research directions.

Summary of Contribution:

The ever-growing data-driven applications such as machine learning and deep learning lead to dramatic increase in computation and communication in today’s systems. Traditional technologies and architectures suffer from the mismatch in computing and network system performance, which have shifted performance bottleneck from the compute to the network. Silicon photonics provide high-bandwidth and reconfigurable interconnects that can address the network challenges presented in datacenter and HPC systems.

In Chapter 2, we first presented FPGA-controlled silicon photonic interconnects, and demonstrated open-loop and closed-loop control over cascaded microring-based switches. The achieved optical unicast, multicast, and multiwavelength-select functionalities are essential for the novel architectures presented in the following chapters to improve system performance in deep learning training.

In Chapter 3, we proposed a photonic switched optically connected memory architecture to address the memory challenges in deep learning. The proposed system architecture utilizes a “lite” (de)serialization scheme for memory transfers via optical links to avoid network overheads

and supports the dynamic allocation of remote memories to local processing systems. We built an experimental testbed with a processing system and two remote memory nodes using silicon photonic switch fabrics and evaluated the system performance. An end-to-end reconfiguration time at millisecond-scale is achieved. The collective results and existing high-bandwidth optical I/Os show the potential of integrating the photonic switched optically connected memory to state-of-the-art processing systems.

In Chapter 4, we demonstrated a silicon photonic switched architecture. The system architecture leverages silicon photonic switch-enabled server regrouping using bandwidth steering to tackle the network challenges and accelerate distributed deep learning training. The proposed system architecture utilizes a highly integrated OS-based SiP switch control scheme to reduce implementation complexity. We built an experimental testbed with a SiP switch-enabled reconfigurable fat tree topology and evaluated the network performance. The large-scale simulation results also show that server regrouping can deliver flow throughput improvement for a tapered fat tree when higher-layer bandwidth steering is employed. The collective results show the potential of integrating SiP switches into datacenter and HPC systems to accelerate distributed deep learning training.

In Chapter 5, we discussed two silicon photonic-enabled disaggregated system architectures for deep learning. The first architecture utilizes optical switches to defragment the computing resources in disaggregated systems. Increased GPU utilization and accelerated network performance are observed. The second architecture is called SiPAC - silicon photonic accelerated compute cluster. It uses silicon photonic multi-wavelength selective switches to achieve a high-bandwidth multi-dimensional all-to-all topology to speed up training communication collectives. The collective results show communication time improvement over current state-of-the-art compute clusters at scale.

In Chapter 6, we reported the development of O-NIC module for the FLEET project. The overall packaging plan for the full assembly was presented. We showed the design of the PIC chip, and the progress on developing the O-NIC PCB. Test packages for design and function verification

were demonstrated and the interface to the host through PCIe was discussed. The design methodologies and approaches show the path towards integrating of silicon photonic transceivers into high performance systems.

Recommendations for Future Work:

Following the work in this dissertation, there are a range of research work and opportunities to further show the potential of silicon photonic interconnects before widely deployed.

A direct step moving forward is to continue the development of the FLEET O-NIC module. The target module and its packaging plan are compatible to be plugged into FPGA board with HBMs. This provides more bandwidth to memory compared to the prototype system shown in Chapter 2. An FPGA can be used to emulate a compute unit and improved performance can be achieved for better demonstrating optically connected memory for deep learning. Additionally, broadband silicon photonic switches, such as MZI based and MEMS-actuated switches, can be incorporated for system demonstrations.

Another usage of the FLEET package is for the disaggregated architecture discussed in Section 5.1. FPGAs can be used to emulate the GPUs as well as the aggregator switches. With higher bandwidth connectivity between the emulated compute and the aggregator switches, the potential results can be more convincing for researchers and providers to continue the development of optically disaggregated systems.

Lastly, more efforts can be made in tunable multi-wavelength selective switches. For example, a dual-ring based filter can drop multiple wavelengths within a FSR. By tuning the coupling coefficients in the dual ring system, bandwidth steering can be realized in the crossbar switch as described in Section 5.2. A flexible SiPAC architecture and better performance can be achieved for more communication collectives in deep learning.

References

- [1] F. J. Ordóñez and D. Roggen, “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition,” *Sensors*, vol. 16, no. 1, p. 115, 2016.
- [2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [3] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [4] R. Raina, A. Madhavan, and A. Y. Ng, “Large-scale deep unsupervised learning using graphics processors,” in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 873–880.
- [5] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, “A dynamically configurable coprocessor for convolutional neural networks,” in *Proceedings of the 37th annual international symposium on Computer architecture*, 2010, pp. 247–257.
- [6] J. Hestness *et al.*, “Deep learning scaling is predictable, empirically,” *arXiv preprint arXiv:1712.00409*, 2017.
- [7] J. Hestness, N. Ardalani, and G. Diamos, “Beyond human-level accuracy: Computational challenges in deep learning,” in *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, 2019, pp. 1–14.
- [8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
- [10] C. Raffel *et al.*, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *arXiv preprint arXiv:1910.10683*, 2019.
- [11] T. Brown *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [12] J. Kaplan *et al.*, “Scaling laws for neural language models,” *arXiv preprint arXiv:2001.08361*, 2020.

- [13] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, “Transformer-xl: Attentive language models beyond a fixed-length context,” *arXiv preprint arXiv:1901.02860*, 2019.
- [14] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, “Megatron-lm: Training multi-billion parameter language models using model parallelism,” *arXiv preprint arXiv:1909.08053*, 2019.
- [15] S. Smith *et al.*, “Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model,” *arXiv preprint arXiv:2201.11990*, 2022.
- [16] W. Fedus, B. Zoph, and N. Shazeer, “Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity,” *arXiv preprint arXiv:2101.03961*, 2021.
- [17] D. Mudigere *et al.*, “Software-hardware co-design for fast and scalable training of deep learning recommendation models,” *arXiv preprint arXiv:2104.05158*, 2021.
- [18] Y. Huang *et al.*, “Gpipe: Efficient training of giant neural networks using pipeline parallelism,” *Advances in neural information processing systems*, vol. 32, 2019.
- [19] P. Goyal *et al.*, “Accurate, large minibatch sgd: Training imagenet in 1 hour,” *arXiv preprint arXiv:1706.02677*, 2017.
- [20] M. Li *et al.*, “Scaling distributed machine learning with the parameter server,” in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, 2014, pp. 583–598.
- [21] N. Shazeer *et al.*, “Mesh-tensorflow: Deep learning for supercomputers,” *Advances in neural information processing systems*, vol. 31, 2018.
- [22] A. Harlap *et al.*, “Pipedream: Fast and efficient pipeline parallel dnn training,” *arXiv preprint arXiv:1806.03377*, 2018.
- [23] A. Canziani, A. Paszke, and E. Culurciello, “An analysis of deep neural network models for practical applications,” *arXiv preprint arXiv:1605.07678*, 2016.
- [24] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.
- [25] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *arXiv preprint arXiv:1609.04836*, 2016.

- [26] U. Gupta *et al.*, “The architectural implications of facebook’s dnn-based personalized recommendation,” in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, 2020, pp. 488–501.
- [27] M. Naumov *et al.*, “Deep learning training in facebook data centers: Design of scale-up and scale-out systems,” *arXiv preprint arXiv:2003.09518*, 2020.
- [28] G. Michelogiannakis, K. Z. Ibrahim, J. Shalf, J. J. Wilke, S. Knight, and J. P. Kenny, “Aphid: Hierarchical task placement to enable a tapered fat tree topology for lower power and cost in hpc networks,” in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, IEEE, 2017, pp. 228–237.
- [29] D. Narayanan *et al.*, “Efficient large-scale language model training on gpu clusters using megatron-lm,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–15.
- [30] *Nvidia a100 tensor core gpu*, <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf>.
- [31] *Nvidia v100 tensor core gpu*, <https://images.nvidia.com/content/technologies/volta/pdf/volta-v100-datasheet-update-us-1165301-r5.pdf>.
- [32] *Nvidia tesla p100 gpu accelerator*, <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-p100/pdf/nvidia-tesla-p100-datasheet.pdf>.
- [33] *Nvidia tesla m40 gpu accelerator*, https://images.nvidia.com/content/tesla/pdf/78071_Tesla_M40_24GB_Print_Datasheet_LR.PDF.
- [34] *Nvidia connectx-7 400g*, <https://nvdam.widen.net/s/m6pt7j5rlb/networking-datasheet-infiniband-connectx-7-ds---1779005>.
- [35] *Nvidia connectx-6*, <https://nvdam.widen.net/s/5j7xtzqfxd/connectx-6-infiniband-datasheet-1987500-r2>.
- [36] *Nvidia connectx-5*, <https://nvdam.widen.net/s/pkxbnmbgkh/networking-infiniband-datasheet-connectx-5-2069273>.
- [37] *Nvidia connectx-4*, <https://network.nvidia.com/files/doc-2020/pb-connectx-4-lx-en-card.pdf>.
- [38] J Kim *et al.*, “1100 x 1100 port mems-based optical crossconnect with 4-db maximum loss,” *IEEE Photonics Technology Letters*, vol. 15, no. 11, pp. 1537–1539, 2003.

- [39] A. N. Dames, *Beam steering optical switch*, US Patent 7,389,016, 2008.
- [40] B. Robertson *et al.*, “Demonstration of multi-casting in a 1×9 lcos wavelength selective switch,” *Journal of lightwave technology*, vol. 32, no. 3, pp. 402–410, 2013.
- [41] C.-T. Lea, “A scalable awgr-based optical switch,” *Journal of lightwave technology*, vol. 33, no. 22, pp. 4612–4621, 2015.
- [42] Q. Cheng, S. Rumley, M. Bahadori, and K. Bergman, “Photonic switching in high performance datacenters,” *Optics express*, vol. 26, no. 12, pp. 16 022–16 043, 2018.
- [43] Q. Cheng, M. Bahadori, M. Glick, S. Rumley, and K. Bergman, “Recent advances in optical technologies for data centers: A review,” *Optica*, vol. 5, no. 11, pp. 1354–1370, 2018.
- [44] K. Suzuki *et al.*, “Low-loss, low-crosstalk, and large-scale optical switch based on silicon photonics,” *Journal of Lightwave Technology*, vol. 38, no. 2, pp. 233–239, 2020.
- [45] H. Subbaraman *et al.*, “Recent advances in silicon-based passive and active optical interconnects,” *Optics express*, vol. 23, no. 3, pp. 2487–2511, 2015.
- [46] W. Bogaerts and L. Chrostowski, “Silicon photonics circuit design: Methods, tools and challenges,” *Laser & Photonics Reviews*, vol. 12, no. 4, p. 1 700 237, 2018.
- [47] K. Suzuki *et al.*, “Low-insertion-loss and power-efficient 32×32 silicon photonics switch with extremely high- Δ silica plc connector,” *Journal of Lightwave Technology*, vol. 37, no. 1, pp. 116–122, 2018.
- [48] K. Suzuki *et al.*, “Nonduplicate polarization-diversity 32×32 silicon photonics switch based on a sin/si double-layer platform,” *Journal of Lightwave Technology*, vol. 38, no. 2, pp. 226–232, 2020.
- [49] T. Chu, L. Qiao, W. Tang, D. Guo, and W. Wu, “Fast, high-radix silicon photonic switches,” in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, IEEE, 2018, pp. 1–3.
- [50] L. Lu *et al.*, “ 16×16 non-blocking silicon optical switch based on electro-optic mach-zehnder interferometers,” *Optics express*, vol. 24, no. 9, pp. 9295–9307, 2016.
- [51] L. Qiao, W. Tang, and T. Chu, “ 32×32 silicon electro-optic switch with built-in monitors and balanced-status units,” *Scientific Reports*, vol. 7, no. 1, pp. 1–7, 2017.
- [52] Q. Cheng, A. Wonfor, J. Wei, R. Penty, and I. White, “Demonstration of the feasibility of large-port-count optical switching using a hybrid mach–zehnder interferometer–semiconductor optical amplifier switch module in a recirculating loop,” *Optics letters*, vol. 39, no. 18, pp. 5244–5247, 2014.

- [53] P. DasMahapatra, R. Stabile, A. Rohit, and K. A. Williams, "Optical crosspoint matrix using broadband resonant switches," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 20, no. 4, pp. 1–10, 2014.
- [54] Y. Huang *et al.*, "Multi-stage 8×8 silicon photonic switch based on dual-microring switching elements," *Journal of Lightwave Technology*, vol. 38, no. 2, pp. 194–201, 2019.
- [55] Q. Cheng *et al.*, "Ultralow-crosstalk, strictly non-blocking microring-based optical switch," *Photonics Research*, vol. 7, no. 2, pp. 155–161, 2019.
- [56] N. Sherwood-Droz *et al.*, "Optical 4×4 hitless silicon router for optical networks-on-chip (noc)," *Optics express*, vol. 16, no. 20, pp. 15 915–15 922, 2008.
- [57] A. Gazman *et al.*, "Software-defined control-plane for wavelength selective unicast and multicast of optical data in a silicon photonic platform," *Optics express*, vol. 25, no. 1, pp. 232–242, 2017.
- [58] K. Padmaraju, D. F. Logan, T. Shiraishi, J. J. Ackert, A. P. Knights, and K. Bergman, "Wavelength locking and thermally stabilizing microring resonators using dithering signals," *Journal of Lightwave Technology*, vol. 32, no. 3, pp. 505–512, 2013.
- [59] J. A. Cox, A. L. Lentine, D. C. Trotter, and A. L. Starbuck, "Control of integrated microresonator wavelength via balanced homodyne locking," *Optics express*, vol. 22, no. 9, pp. 11 279–11 289, 2014.
- [60] T. J. Seok, K. Kwon, J. Henriksson, J. Luo, and M. C. Wu, "Wafer-scale silicon photonic switches beyond die size limit," *Optica*, vol. 6, no. 4, pp. 490–494, 2019.
- [61] S. Han, T. J. Seok, K. Yu, N. Quack, R. S. Muller, and M. C. Wu, "Large-scale polarization-insensitive silicon photonic mems switches," *Journal of Lightwave Technology*, vol. 36, no. 10, pp. 1824–1830, 2018.
- [62] S. Han *et al.*, " 32×32 silicon photonic mems switch with gap-adjustable directional couplers fabricated in commercial cmos foundry," *Journal of Optical Microsystems*, vol. 1, no. 2, p. 024 003, 2021.
- [63] N. C. Abrams *et al.*, "Silicon photonic 2.5 d multi-chip module transceiver for high-performance data centers," *Journal of Lightwave Technology*, vol. 38, no. 13, pp. 3346–3357, 2020.
- [64] D. Che, A. Li, X. Chen, Q. Hu, Y. Wang, and W. Shieh, "Stokes vector direct detection for short-reach optical communication," *Optics letters*, vol. 39, no. 11, pp. 3110–3113, 2014.

- [65] J. K. Perin, A. Shastri, and J. M. Kahn, "Design of low-power dsp-free coherent receivers for data center links," *Journal of Lightwave Technology*, vol. 35, no. 21, pp. 4650–4662, 2017.
- [66] M. Morsy-Osman *et al.*, "Dsp-free 'coherent-lite' transceiver for next generation single wavelength optical intra-datacenter interconnects," *Optics Express*, vol. 26, no. 7, pp. 8890–8903, 2018.
- [67] J. Cheng, C. Xie, M. Tang, and S. Fu, "A low-complexity adaptive equalizer for digital coherent short-reach optical transmission systems," in *2019 Optical Fiber Communications Conference and Exhibition (OFC)*, IEEE, 2019, pp. 1–3.
- [68] M. Wade *et al.*, "Teraphy: A chiplet technology for low-power, high-bandwidth in-package optical i/o," *IEEE Micro*, vol. 40, no. 2, pp. 63–71, 2020.
- [69] A. Rizzo *et al.*, "Integrated kerr frequency comb-driven silicon photonic transmitter," *arXiv preprint arXiv:2109.10297*, 2021.
- [70] B. Y. Kim *et al.*, "Turn-key, high-efficiency kerr comb source," *Optics letters*, vol. 44, no. 18, pp. 4475–4478, 2019.
- [71] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2010 Conference*, 2010, pp. 339–350.
- [72] G. Porter *et al.*, "Integrating microsecond circuit switching into the data center," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 447–458, 2013.
- [73] M. Ghobadi *et al.*, "Projector: Agile reconfigurable data center interconnect," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 216–229.
- [74] K. Wen *et al.*, "Flexfly: Enabling a reconfigurable dragonfly through silicon photonics," in *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, IEEE, 2016, pp. 166–177.
- [75] P. Duthie and M. Wale, "16* 16 single chip optical switch array in lithium niobate," *Electronics letters*, vol. 27, no. 14, pp. 1265–1266, 1991.
- [76] S. Sohma, T. Watanabe, N. Ooba, M. Itoh, T. Shibata, and H. Takahashi, "Silica-based plc type 32 x 32 optical matrix switch," in *2006 European Conference on Optical Communications*, IEEE, 2006, pp. 1–2.
- [77] M. Bahadori *et al.*, "Crosstalk penalty in microring-based silicon photonic interconnect systems," *Journal of Lightwave Technology*, vol. 34, no. 17, pp. 4043–4052, 2016.

- [78] C. Browning, K. Shi, A. D. Ellis, and L. P. Barry, “Optical burst-switched ssb-ofdm using a fast switching sg-dbr laser,” *Journal of Optical Communications and Networking*, vol. 5, no. 9, pp. 994–1000, 2013.
- [79] A. E.-J. Lim *et al.*, “Review of silicon photonics foundry efforts,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 20, no. 4, pp. 405–416, 2013.
- [80] D. Y. Lee *et al.*, “Error-free operation of a polarization-insensitive 4λ x 25 gbps silicon photonic wdm receiver with closed-loop thermal stabilization of si microrings,” *Optics Express*, vol. 24, no. 12, pp. 13 204–13 209, 2016.
- [81] P. Dong *et al.*, “Simultaneous wavelength locking of microring modulator array with a single monitoring signal,” *Optics express*, vol. 25, no. 14, pp. 16 040–16 046, 2017.
- [82] A. Gazman, C. Browning, Z. Zhu, L. R. Barry, and K. Bergman, “Automated thermal stabilization of cascaded silicon photonic ring resonators for reconfigurable wdm applications,” in *2017 European Conference on Optical Communication (ECOC)*, IEEE, 2017, pp. 1–3.
- [83] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar, and S. W. Keckler, “Vdnn: Virtualized deep neural networks for scalable, memory-efficient neural network design,” in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, 2016, pp. 1–13.
- [84] A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks,” *arXiv preprint arXiv:1404.5997*, 2014.
- [85] Y. Kwon and M. Rhu, “Beyond the memory wall: A case for memory-centric hpc system for deep learning,” in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, 2018, pp. 148–161.
- [86] A. Eisenman *et al.*, “Bandana: Using non-volatile memory for storing deep learning models,” *Proceedings of Machine Learning and Systems*, vol. 1, pp. 40–52, 2019.
- [87] G. Zervas, H. Yuan, A. Saljoghei, Q. Chen, and V. Mishra, “Optically disaggregated data centers with minimal remote memory latency: Technologies, architectures, and resource allocation,” *Journal of Optical Communications and Networking*, vol. 10, no. 2, A270–A285, 2018.
- [88] K. Bergman, J. Shalf, G. Michelogiannakis, S. Rumley, L. Dennison, and M. Ghobadi, “Pine: An energy efficient flexibly interconnected photonic data center architecture for extreme scalability,” in *2018 IEEE Optical Interconnects Conference (OI)*, IEEE, 2018, pp. 25–26.

- [89] Y. Yan *et al.*, “All-optical programmable disaggregated data centre network realized by fpga-based switch and interface card,” *Journal of Lightwave Technology*, vol. 34, no. 8, pp. 1925–1932, 2016.
- [90] T. Shiraishi *et al.*, “A reconfigurable and redundant optically-connected memory system using a silicon photonic switch,” in *OFC 2014*, IEEE, 2014, pp. 1–3.
- [91] D. Brunina, C. P. Lai, A. S. Garg, and K. Bergman, “Building data centers with optically connected memory,” *Journal of Optical Communications and Networking*, vol. 3, no. 8, A40–A48, 2011.
- [92] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [93] J. Donahue *et al.*, “Long-term recurrent convolutional networks for visual recognition and description,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.
- [94] A. G. Howard *et al.*, “Mobilenets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.
- [95] *Amba axi4*, <https://www.xilinx.com/products/intellectual-property/axi.html>.
- [96] *Axi chip2chip*, <https://www.xilinx.com/products/intellectual-property/axi-chip2chip.html>.
- [97] *Aurora 64b/66b*, <https://www.xilinx.com/products/intellectual-property/aurora64b66b.html>.
- [98] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv preprint arXiv:1811.03378*, 2018.
- [99] R. Konoike *et al.*, “Soa-integrated silicon photonics switch and its lossless multistage transmission of high-capacity wdm signals,” *Journal of lightwave technology*, vol. 37, no. 1, pp. 123–130, 2018.
- [100] K. Clark *et al.*, “Sub-nanosecond clock and data recovery in an optically-switched data centre network,” in *2018 European Conference on Optical Communication (ECOC)*, IEEE, 2018, pp. 1–3.
- [101] *Nvidia titan rtx is here*, <https://www.nvidia.com/en-us/deep-learning-ai/products/titan-rtx/>.

- [102] B. G. Lee and N. Dupuis, “Silicon photonic switch fabrics: Technology and architecture,” *Journal of Lightwave Technology*, vol. 37, no. 1, pp. 6–20, 2018.
- [103] *Amazon ebs features*, <https://aws.amazon.com/ebs/features/>.
- [104] L. Alloatti *et al.*, “100 ghz silicon–organic hybrid modulator,” *Light: Science & Applications*, vol. 3, no. 5, e173–e173, 2014.
- [105] Y. Salamin *et al.*, “100 ghz plasmonic photodetector,” *ACS photonics*, vol. 5, no. 8, pp. 3291–3297, 2018.
- [106] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf, “Reconfigurable hybrid interconnection for static and dynamic scientific applications,” in *Proceedings of the 4th international conference on Computing frontiers*, 2007, pp. 183–194.
- [107] N. Hamedazimi *et al.*, “Firefly: A reconfigurable wireless data center fabric using free-space optics,” in *Proceedings of the 2014 ACM conference on SIGCOMM*, 2014, pp. 319–330.
- [108] W. M. Mellette *et al.*, “Rotornet: A scalable, low-complexity, optical datacenter network,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 267–280.
- [109] H. Ballani *et al.*, “Sirius: A flat datacenter network with nanosecond optical switching,” in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 782–797.
- [110] M. Y. Teh, Z. Wu, and K. Bergman, “Flexspander: Augmenting expander networks in high-performance systems with optical bandwidth steering,” *Journal of Optical Communications and Networking*, vol. 12, no. 4, B44–B54, 2020.
- [111] T.-N. Truong and R. Takano, “Hybrid electrical/optical switch architectures for training distributed deep learning in large-scale,” *IEICE TRANSACTIONS on Information and Systems*, vol. 104, no. 8, pp. 1332–1339, 2021.
- [112] Y. Lu, H. Gu, X. Yu, and P. Li, “X-nest: A scalable, flexible, and high-performance network architecture for distributed machine learning,” *Journal of Lightwave Technology*, vol. 39, no. 13, pp. 4247–4254, 2021.
- [113] G. Michelogiannakis *et al.*, “Bandwidth steering in hpc using silicon nanophotonics,” in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019, pp. 1–25.

- [114] Z. Zhu, S. Yan, M. S. Glick, M. Y. Teh, and K. Bergman, “Silicon photonic switch-enabled server regrouping using bandwidth steering for distributed deep learning training,” in *Optical Fiber Communication Conference*, Optical Society of America, 2021, Th5H–3.
- [115] *Bringing hpc techniques to deep learning*, <https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce/>.
- [116] Z. Zhu *et al.*, “Photonic switched optically connected memory: An approach to address memory challenges in deep learning,” *Journal of Lightwave Technology*, vol. 38, no. 10, pp. 2815–2825, 2020.
- [117] Y. Shen, S. Rumley, K. Wen, Z. Zhu, A. Gazman, and K. Bergman, “Accelerating of high performance data centers using silicon photonic switch-enabled bandwidth steering,” in *2018 European Conference on Optical Communication (ECOC)*, IEEE, 2018, pp. 1–3.
- [118] *Imagenette*, <https://github.com/fastai/imagenette>.
- [119] *Netbench*, <https://github.com/ndal-eth/netbench>.
- [120] K.-T. Foerster, M. Ghobadi, and S. Schmid, “Characterizing the algorithmic complexity of reconfigurable data center architectures,” in *Proceedings of the 2018 Symposium on Architectures for Networking and Communications Systems*, 2018, pp. 89–96.
- [121] M. Jeon, S. Venkataraman, A. Phanishayee, J. Qian, W. Xiao, and F. Yang, “Analysis of {large-scale}{multi-tenant}{gpu} clusters for {dnn} training workloads,” in *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 2019, pp. 947–960.
- [122] M. Glick *et al.*, “Pine: Photonic integrated networked energy efficient datacenters (enlitened program),” *Journal of Optical Communications and Networking*, vol. 12, no. 12, pp. 443–456, 2020.
- [123] J. Shalf *et al.*, “Photonic memory disaggregation in datacenters,” in *Photonics in Switching and Computing*, Optical Society of America, 2020, PsW1F–5.
- [124] P. X. Gao *et al.*, “Network requirements for resource disaggregation,” in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016, pp. 249–264.
- [125] *Nvidia dgx superpod*, <https://images.nvidia.com/aem-dam/Solutions/Data-Center/gated-resources/nvidia-dgx-superpod-a100.pdf>.
- [126] *Google cloud tpu*, <https://cloud.google.com/tpu>.
- [127] S. Shi, Q. Wang, and X. Chu, “Performance modeling and evaluation of distributed deep learning frameworks on gpus,” in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and*

Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), IEEE, 2018, pp. 949–957.

- [128] Lawrence livermore national laboratory, “sierra”, <https://hpc.llnl.gov/hardware/compute-platforms/sierra>, journal=Sierra | HPC @ LLNL.
- [129] N. Dryden *et al.*, “Aluminum: An asynchronous, gpu-aware communication library optimized for large-scale training of deep neural networks on hpc systems,” Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), Tech. Rep., 2018.
- [130] *Teraphy*, 2021.
- [131] L. Liu, Q. Jin, D. Wang, H. Yu, G. Sun, and S. Luo, “Psnet: Reconfigurable network topology design for accelerating parameter server architecture based distributed machine learning,” *Future Generation Computer Systems*, vol. 106, pp. 320–332, 2020.
- [132] M. Khani *et al.*, “Sip-ml: High-bandwidth optical network interconnects for machine learning training,” in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 657–675.
- [133] W. Wang *et al.*, “Topoopt: Optimizing the network topology for distributed dnn training,” *arXiv preprint arXiv:2202.00433*, 2022.
- [134] C. Guo *et al.*, “Bcube: A high performance, server-centric network architecture for modular data centers,” in *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*, 2009, pp. 63–74.
- [135] L. N. Bhuyan and D. P. Agrawal, “Generalized hypercube and hyperbus structures for a computer network,” *IEEE Transactions on computers*, vol. 33, no. 04, pp. 323–333, 1984.
- [136] A. Novick *et al.*, “Error-free kerr comb-driven sip microdisk transmitter,” in *2021 Conference on Lasers and Electro-Optics (CLEO)*, 2021, pp. 1–2.
- [137] *Dgx-a100 system user guide*, <https://docs.nvidia.com/dgx/pdf/dgxa100-user-guide.pdf>.
- [138] A. Li *et al.*, “Evaluating modern gpu interconnect: Pcie, nvlink, nv-sli, nvswitch and gpudirect,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 1, pp. 94–110, 2019.
- [139] *Introducing 200g hdr infiniband solutions*, <https://network.nvidia.com/sites/default/files/doc-2020/wp-introducing-200g-hdr-infiniband-solutions.pdf>.

- [140] *Darpa fastnics program*, <https://www.darpa.mil/program/fast-network-interface-cards>.
- [141] F. Douglass *et al.*, “Fleet—fast lanes for expedited execution at 10 terabits: Program overview,” *IEEE Internet Computing*, vol. 25, no. 3, pp. 79–87, 2021.
- [142] *Onet2804tlp*, <https://www.ti.com/product/ONET2804TLP>.