JOURNAL OF Optical Communications and Networking

# Flexible silicon photonic architecture for accelerating distributed deep learning

Zhenguo Wu,* Liang Yuan Dai, Yuyang Wang, Songli Wang, and Keren Bergman

*Department of Electrical Engineering, Columbia University, 500 W 120th St., New York, New York 10027, USA*
**zw2542@columbia.edu*

*Received 3 July 2023; revised 23 October 2023; accepted 23 October 2023; published 9 January 2024*

**The increasing size and complexity of deep learning (DL) models have led to the wide adoption of distributed training methods in datacenters (DCs) and high-performance computing (HPC) systems. However, communication among distributed computing units (CUs) has emerged as a major bottleneck in the training process. In this study, we propose Flex-SiPAC, a flexible silicon photonic accelerated compute cluster designed to accelerate multi-tenant distributed DL training workloads. Flex-SiPAC takes a co-design approach that combines a silicon photonic hardware platform with a tailored collective algorithm, optimized to leverage the unique physical properties of the architecture. The hardware platform integrates a novel wavelength-reconfigurable transceiver design and a micro-resonator-based wavelength-reconfigurable switch, enabling the system to achieve flexible bandwidth steering in the wavelength domain. The collective algorithm is designed to support reconfigurable topologies, enabling efficient all-reduce communications that are commonly used in DL training. The feasibility of the Flex-SiPAC architecture is demonstrated through two testbed experiments. First, an optical testbed experiment demonstrates the flexible routing of wavelengths by shuffling an array of input wavelengths using a custom-designed spatial-wavelength selective switch. Second, a four-GPU testbed running two DL workloads shows a 23% improvement in job completion time compared to a similarly sized leaf-spine topology. We further evaluate Flex-SiPAC using large-scale simulations, which show that Flex-SiPAC is able to reduce the communication time by 26% to 29% compared to state-of-the-art compute clusters under representative collective operations.**

https://doi.org/10.1364/JOCN.497372

## 1. INTRODUCTION

Recent AI advancements have been largely driven by the emergence of large-scale deep learning (DL) models. Notably, large language models (LLMs), such as ChatGPT, have demonstrated significant progress across various applications. Their increasing model size and dataset requirements demand distributed deep learning (DDL) solutions, which partition and distribute training models or datasets across clusters of computing units (CUs). As Fig. 1 shows, the reported number of CUs (e.g., GPUs, TPUs, or NPUs) employed in training representative DL models exhibits an exponential growth over the years. This trend is expected to continue into the future (although not yet formally reported, the projected number of GPUs for the next generation of ChatGPT is expected to surpass 30,000 [14]) [15,16]. In state-of-the-art accelerator systems, the bandwidth discrepancy between intra-cluster interconnects and inter-cluster networks has created communication bottlenecks for DDL, severely limiting training efficiency [1].

To mitigate these challenges, an existing approach is to provide uniformly high bandwidth among CUs. For instance,

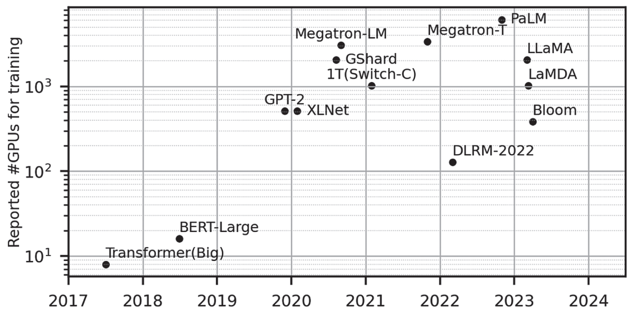

**Fig. 1.** Reported numbers of GPUs utilized in training major models from 2016 to 2023 [1–13].

the Nvidia DGX-H100 SuperPOD system uses high-speed NVSwitches and NVLinks with up to 900 GB/s aggregated bandwidth to connect GPUs both within and across compute nodes. However, these high-speed electronic switches and links incur high cost and energy consumption and are distance limited [17]. To scale the size of the training cluster beyond the limit of the electrical wires, the use of slower interconnects (e.g., 400 Gb/s InfiniBand fabric) becomes inevitable among these groups, which again introduces bandwidth discrepancy at group boundaries. This issue is further compounded when multi-tenant jobs are mapped non-uniformly onto the compute clusters, leading to bandwidth waste in idle links, calling for bandwidth reconfigurability in addition to high capacity.

To address the bandwidth challenges associated with trainings in accelerator clusters, we previously introduced the SiPAC architecture at OFC'23 [18,19]. SiPAC exploits multi-wavelength selective switching and high-bandwidth DWDM links to emulate a multi-dimensional all-to-all topology in the optical domain. This topology provides high-bandwidth direct paths for DDL collective operations, which also exhibit multi-dimensional communication patterns. The co-designed collective algorithm builds on this hardware multi-casting capability to achieve both latency and bandwidth efficiency.

In this work, we introduce Flex-SiPAC, a flexible silicon photonic accelerated compute architecture. Building upon the prior SiPAC architecture, Flex-SiPAC introduces bandwidth reconfiguration in the wavelength domain to enhance network agility in a multi-tenant compute cluster where jobs are heterogeneous in both size and mapping. To flexibly route wavelengths, we employ (1) comb-driven DWDM transceivers with multiple reconfigurable output ports that can flexibly split the comb wavelengths and (2) micro-resonator-based spatial-wavelength selective switches (SWSSs) that together enable flexible traffic aware topology optimization. Leveraging this wavelength reconfiguration primitive, we also propose a co-designed topology-aware all-reduce collective communication algorithm that can efficiently utilize the dynamically allocated wavelengths. This work augments the architecture presented earlier with reconfigurable capabilities and presents a comprehensive analysis of this proposed design. The major contributions of this work are summarized as follows:

- **Wavelength Reconfigurable Interconnect Fabric:** We demonstrate the integration of reconfigurable capabilities into both the transceiver and switch, enabling the construction of a fully wavelength reconfigurable interconnect fabric for the compute cluster. We report a testbed experiment using a WSS where a variable number of input wavelengths are routed to different output ports. The experimental results demonstrate the feasibility of flexible and high-bandwidth communication enabled by the reconfigurable WSS in the Flex-SiPAC architecture.
- **Traffic-Aware Wavelength Reconfiguration:** We show how to leverage the wavelength reconfigurable interconnect fabric to achieve traffic-aware topology reconfiguration for multi-tenant DL traffic workloads. We propose a heuristic approach to solve the integer linear programming (ILP) optimization problem formulated in the context of the bandwidth steering algorithm. We report small-scale system-level testbed results that show a 23% performance improvement relative to a similarly sized leaf-spine topology on DDL workloads.
- **Topology-Aware All-Reduce Collective Algorithm Co-design:** We present a co-designed all-reduce collective communication algorithm, Flex-SiPCO, that leverages the reconfigured bandwidth of Flex-SiPAC to reduce bandwidth waste and achieve higher communication efficiency. To evaluate the performance of our proposed Flex-SiPAC-Flex-SiPCO co-design, we conduct detailed packet-level simulations on representative DDL workloads. Large-scale simulation results show that our topology-algorithm co-design improves the communication time by a factor of 26% to 29% compared to the state-of-the-art accelerator clusters.

## 2. BACKGROUND AND RELATED WORK

In this section, we provide an overview of the state-of-the-art accelerator architectures developed under both commercial and research settings and provide background on the collective communication patterns commonly observed in DDL workloads.

### A. Network Architectures for Accelerator Clusters

DL workloads demand specialized accelerators to achieve high-performance computations while relying on efficient networking solutions to scale the size of the training. In this section, we describe the state-of-the-art accelerator systems and discuss ongoing research efforts focused on enhancing networking performance through the integration of optical switching technologies.

#### *1. Static Network Architectures*

To meet the high-bandwidth and low-latency demand of the data-movement intensive workload, various commercial architectures have been developed to facilitate efficient communication in accelerator clusters. For example, Meta's ZionEX architecture incorporates a custom accelerator equipped with advanced networking capabilities, including a high-bandwidth and flexible intra-node topology. However, the scale-out bandwidth is much lower than the intra-node bandwidth, which limits the efficiency of the collective communications [9]. Similarly, earlier versions of Google's TPU Pod [20] architecture adopted a high-speed intra-pod fabric wired in a 2D-torus topology. Scaling the 2D-torus becomes distance limited as wrap-around links become so long that they have reached the limits of electrical interconnects. As a solution, optical fibers are used for these long links. Another notable architecture is the Nvidia DGX-SuperPOD, which uses specialized electrical switching fabric to facilitate high-speed communication. In earlier releases of this architecture (i.e., prior to DGX-H100 SuperPOD), only GPUs within a server were interconnected using high-speed NVLinks and NVSwitches, while the servers themselves were interconnected using slower InfiniBand fabrics. In summary, the general approach across different state-of-the-art architectures was to provide extremely high bandwidth among CUs in a group. Initially, this approach was effective since most of the workloads could be contained inside a group, but with the increasing scale of DDL model and dataset sizes, research has highlighted the challenges stemming from bandwidth discrepancies between intra-group and inter-group links. Collective operation becomes bottlenecked at the slower inter-server links, resulting in higher communication costs [1]. To address this challenge, Nvidia's latest release, the H100-SuperPOD architecture, introduces an additional layer of NVSwitches and NVLinks, forming an all-NVLink connected topology to provide uniform high bandwidth among GPUs within the reach of the electrical links.

### 2. *Reconfigurable Network Architectures*

Recent research studies have focused on integrating optical switching technologies into DCs to enhance their flexibility [21]. Some examples include using MEMS-based optical circuit switches (OCSs) to address traffic imbalances in DC workloads, which often exhibit temporal and spatial localities [22]. These OCSs enable the dynamic allocation of link bandwidth to traffic hotspots using spatial switching, which involves simultaneously switching all wavelength channels from input to output ports. Another approach involves using fast optical switches [23] or fast tunable transceivers [24] for more rapid bandwidth reconfiguration. However, frequent network reconfiguration could lead to significant overhead in the control plane and has been shown to yield limited benefits in DC settings [25–27].

As the use of accelerator clusters becomes prevalent within DC and HPC systems, this trend toward optical switching has also been extended to accelerator network architectures. In Google's latest TPUv4 Pod [16], OCSs are employed to connect outer wrap-around links for the 3D-torus base blocks, enabling topology engineering for different job mappings and training parallelisms. Additionally, other OCS-based reconfigurable accelerator systems for distributed learning applications have been proposed [28–30]. However, these MEMS-based OCSs lack the capability to selectively route individual wavelengths in the wavelength domain, which restricts switching granularity, particularly in DWDM architectures. The Flex-LIONS architecture uses a combination of arrayed waveguide grating router (AWGR) and Mach–Zehnder switches (MZSs) to enable the reconfiguration of individual wavelengths in the optical domain while maintaining a static all-to-all GPU interconnect [31]. However, scaling the number of CUs in an all-to-all fashion is limited by the finite number of wavelengths per transmitter. To address this limitation, a commonly employed approach is to introduce hierarchy into the system. The SiPAC architecture [18] implements a connection scheme based on the BCube topology but utilizes micro-resonator-based WSSs to route DWDM wavelengths. This approach enables the realization of a multi-dimensional all-to-all network in the optical domain, which enables scaling beyond the limitations of a fully all-to-all network.

### B. Collective Communication in DDL Workloads

DDL relies on collective communication to synchronize the intermediary results of each individual CU within the distributed computing cluster. The specific communication pattern varies depending on the type of operation involved in different parallelization strategies. A dominant collective operation in data parallelism and certain tensor model parallelisms [32] is the all-reduce operation, where each CU has a data buffer containing the outcomes obtained by executing a pointwise calculation on the corresponding index across all CUs [33]. Other collective operations, such as the reduce-scatter, all-gather, can be derived from all-reduce (i.e., all-reduce can be decomposed into reduce-scatter and all-gather) with modifications to the transmission pattern and operations performed. Therefore, we will mainly focus on the all-reduce operation in this study. Other collective operations, such as primitive all-to-all, have been evaluated in the static cases in [18]. Many algorithms for all-reduce have been proposed in the past, including ring-based [34], hierarchical ring-based [35], and mesh-based [36], each with different latency versus bandwidth trade-offs. These algorithms are oblivious to the underlying physical topologies and rely on pre-configured templates designed for specific collective topologies (i.e., ring, hierarchical-ring, and mesh topologies). The traffic pattern formed by the particular collective algorithm is therefore relatively stable across the iterations of the training process. We propose an all-reduce collective algorithm co-designed with the Flex-SiPAC physical topology, leveraging the optical multicasting and wavelength reconfiguration capabilities of our proposed optical hardware.

## 3. Flex-SiPAC ARCHITECTURE

In this section, we describe the Flex-SiPAC design, highlighting the details of the SiP technologies employed in this architecture. All mathematical notations are summarized in Table 1.

### A. Topology Design

The Flex-SiPAC architecture maintains the same link-level connections as the SiPAC [18] architecture, which is based on the BCube physical topology [37]. SiPAC($r$, $l$) is a recursively defined topology where $r$ and $l$ represents the switch radix and the current level/dimension in the topology. The base unit of level $l = 0$ consists of a single $r$-port WSS connecting $r$ disaggregated CUs. According to the recurrence relation, for each level $l > 0$, SiPAC($r$, $l$) is constructed by connecting $r^l$ $r$-port WSSs, each of which connects $r$ SiPAC($r$, $l - 1$) units, as illustrated in Fig. 2(a). Each CU is therefore equipped with $l + 1$ optical ports, which is equivalent to the number of levels in the topology as well as the diameter of the topology. Similar to BCube, CUs in Flex-SiPAC serves both as end hosts and relay nodes. To minimize the impact of traffic relaying, our

**Table 1. List of Mathematical Symbols and Their Respective Descriptions**

| Notation | Description |
|---|---|
| $r$ | Switch radix |
| $l$ | Level index in the Flex-SiPAC topology, $l \in [0,\ L-1]$, where $L = \max(l) + 1$ |
| $w$ | Number of tx/rx wavelengths per transceiver |
| $N$ | Number of CUs in the topology |
| $M$ | Number of WSSs in the topology |
| $P$ | Number of jobs mapped to the topology |
| $U$ | Set of CUs in job $i$: $U = \{u_i\}$, $i \in [0,\ P-1]$ |
| $S$ | Set of WSSs in level $l$: $S = \{s_l\}$, $l \in [0,\ L-1]$ |
| $T$ | Traffic matrix: $T = [t_{ij}] \in \mathbb{R}^{N\times N}$, $t_{ij} = 0$, $\forall i = j$ |
| $\Lambda$ | Number of wavelengths among each CU-pair (through $l$th level WSS): $\Lambda = [\lambda_{ij}^{(l)}] \in \mathbb{R}^{N\times N}$, $\lambda_{ij} = 0$, $\forall i = j$ |

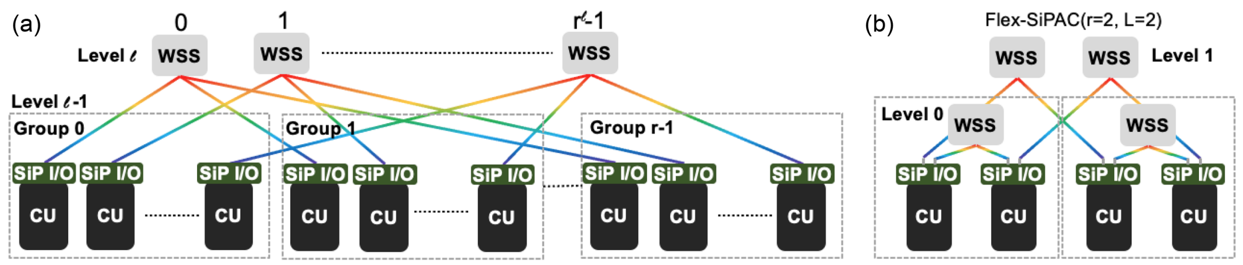

**Fig. 2.** (a) Flex-SiPAC architecture based on the recursive BCube topology [37]. The $l$th level is constructed from $r^l$ $r$-port switches and $r(l-1)$th level units. (b) Example Flex-SiPAC($r=2$, $L=2$) network showing two port connections per CU and full connectivity in level 0.

co-designed collective algorithm (Section 4) enables each CU to communicate only with its immediate neighbors in each time step under uniform job mapping. When transit traffic is required, the algorithm ensures that the transit message traverses the fewest intermediate hops before an operation (i.e., reduction) is performed.

In the Flex-SiPAC architecture, the electronic packet switches (EPSs) are replaced with reconfigurable WSSs, and the comb-source driven transceiver architecture on each CU is redesigned to incorporate reconfigurability at the endpoints. Initially, the wavelengths from each transmitter are uniformly distributed to each output port, and the WSS in each dimension uniformly routes the wavelengths to all directly connected neighbors. It should be noted that current comb-driven laser sources already offer 160 wavelengths [38], and silicon photonic modulators have achieved a data rate of 128 Gb/s per wavelength [39]. Scalable micro-ring resonator (MRR) switch designs have also demonstrated feasibility for port counts up to 128 [40]. However, achieving these numbers altogether requires addressing several challenges, including managing insertion loss, handling temperature variations, engineering the MRR's free spectral range (FSR) to align with the wavelengths, and scaling the switch port count. These benchmarks can provide valuable references for future scalability while acknowledging the associated challenges. Figure 3 illustrates the projected trend for future bandwidth scaling as a function of the switch radix used in the Flex-SiPAC architecture. Under a uniform wavelength distribution, the per-link bandwidth can reach up to 4 Tb/s. Since each CU in a SiPAC($r$, $l$) can reach ($r-1$) direct neighbors via wavelength routing in a single dimension, the CU-to-CU direct bandwidth can reach up to 256 Gb/s when the switch radix is 16. The WSS radix used is

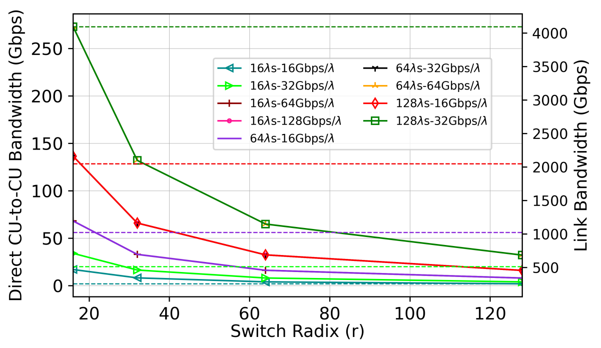

**Fig. 3.** Flex-SiPAC bandwidth scaling under uniform distribution of wavelengths.

typically small since the number of endpoints scales exponentially with the number of levels in the topology. For example, a radix 16 switch could achieve a topology size of 256 for $L=2$ and 4096 for $L=3$. Furthermore, the ability to flexibly tune the wavelengths at both the transceiver and the switch allows us to achieve a much larger design space that is characterized by the entire space under the dotted horizontal curves in Fig. 3 for a given link bandwidth.

## B. SiP Technologies for Flex-SiPAC

### 1. SiP Wavelength Reconfigurable Switches

The WSS proposed in the SiPAC architecture [18] has the ability to route multiple wavelengths from a single input port to each of its directly connected neighbors, creating a high-bandwidth all-to-all optical topology within each dimension. This feature enables efficient optical multicasting, improving the communication efficiency of collective operations. In this work, we enhance this design by incorporating spatial reconfigurability into the WSSs. Our objective is to route a flexible number of incoming wavelengths from any input port to any output port while avoiding coherent crosstalk at each output bus. This approach allows us to achieve similar benefits to previous studies on reconfigurable networks [16,28,29,41–43], but in the wavelength domain with finer granularity. We propose a few options for this switch design. One possible implementation to achieve this functionality is through the use of a double-crosspoint micro-resonator based switch, which can be explored further based on the work in [44]. This implementation has an added advantage in that it can be designed to oversubscribe the number of wavelength-selective cells to accommodate the broadband wavelengths from the reconfigurable transceivers that will be introduced next. Another approach involves replacing regular micro-ring-based resonators in a cross-bar switch with tunable switching cells, where the free-spectral-ranges (FSR) of each switch cell can be adjusted to drop a specific number of wavelengths. The work on these switching devices is ongoing and will be presented in subsequent publications.

### 2. SiP Wavelength Reconfigurable Transceivers

The integration of optical transceiver ports directly onto chip interposers eliminates the need for expensive network interface cards (NICs). In our design, each CU in Flex-SiPAC is equipped with an embedded comb-driven DWDM transceiver, such as the one described in [45]. In addition to providing massively parallel DWDM bandwidths, the transceivers are augmented by introducing wavelength reconfigurability to dynamically route wavelengths across different output ports. By default, the DWDM wavelengths are evenly distributed across fibers that connect to CUs in different dimensions. However, if the traffic pattern demands higher bandwidth than what a single dimension can support, we can configure the transceivers to combine wavelengths from different output ports and transmit them through a single output port.

To achieve this, we incorporate a reconfigurable (de)-interleaving structure after the embedded transceiver that splits the original single fiber output into multiple fiber outputs. Figure 4(a) illustrates an architectural example of such a de-interleaving structure design, where the central broadband de-interleaver is assisted by several balanced Mach–Zehnder interferometers (MZIs) to route wavelengths to different paths. In this $1 \times 2$ transceiver configuration, modulated DWDM wavelengths entering the de-interleaving structure can be configured to travel through different light paths by tuning each of the MZIs. For an evenly split distribution, the wavelengths are routed through the center broadband de-interleaver [Fig. 4(b)]. The specific implementation of this de-interleaving structure can vary based on system requirements. Different wavelength interleaving schemes, such as odd-even interleaving [46] or band-interleaving [47,48], can be employed. When we need to redirect wavelengths from the second output port to the first, we can tune the MZIs accordingly so that all wavelengths are routed through the upper branch, bypassing the broadband de-interleaver. Symmetrical operations can be performed for routing all wavelengths to the second output port. This reconfigurable de-interleaving structure serves as a base unit that can be cascaded into multiple stages, hierarchically achieving more reconfigurable output ports. The number of ports required is equivalent to the number of levels in Flex-SiPAC, which is typically small (e.g., 2 or 3). The combined effect of this tunable transceiver-switch co-design allows us to flexibly reconfigure wavelengths across different dimensions as well as among different endpoints within a single dimension. Ongoing research is being conducted on these reconfigurable transceiver implementations, and the results will be presented in future works.

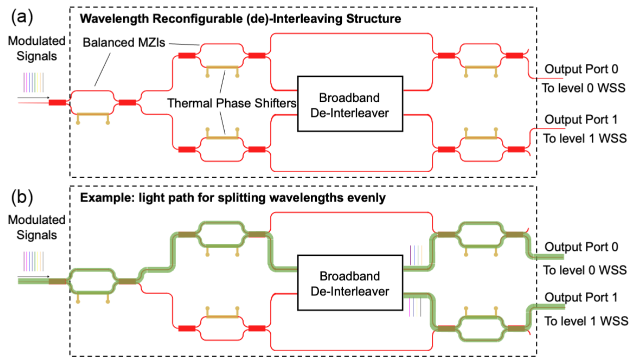

**Fig. 4.** (a) Example of a $1 \times 2$ wavelength reconfigurable transceiver architecture. (b) Evenly distributing incoming modulated wavelengths across the two output ports.

### C. Topology Engineering in the Wavelength Domain

We now formalize the wavelength granularity bandwidth steering algorithm within the context of a Flex-SiPAC network. We begin by assuming that jobs are mapped sequentially onto the cluster, with each job selecting a collective algorithm such as ring, hierarchical-ring, mesh, or SiPCO [19] to carry out the collective operation based on the parallelism of interest. For example, when four jobs of size four are sequentially mapped onto a Flex-SiPAC ($r = 4$, $l = 1$) using the mesh algorithm, each sub-traffic matrix forms a full mesh across the four CUs under the same level 0 WSS. However, when two jobs of size eight are assigned, each sub-traffic matrix establishes a full mesh across the eight CUs, spanning two levels of WSSs. Given the sub-traffic matrix formed by the selected collective algorithm in each job, we construct a full topology traffic matrix $T = [t_{ij}]$ by aggregating these sub-traffic matrices. These initial traffic matrices used for topology optimization are simplified representations of real traffic matrices to redirect bandwidth that initially spans CUs across multiple jobs to within CUs belonging to the same job. In the case of Flex-SiPAC, we pick SiPCO to restrict communication to directly connected neighbors. This ensures that $t_{ij} = 0, \forall i, j : \lambda_{ij} = 0$. We then filter out any traffic that crosses CUs between different jobs. Our objective is twofold: (1) determining how the wavelengths from the input port of the transceiver should be allocated to each output port and (2) configuring the switch states in each dimension to maximize network throughput. We describe our desired topology in the wavelength domain as $\Lambda = [\lambda_{ij}]$. To align the topology with the traffic matrix, we aim to allocate wavelengths between CUs in proportion to the volume of traffic exchanged between them. To achieve this, we employ an ILP formulation as expressed in Eq. (1).

The decision variable $\lambda_{ij}$ represents the number of wavelengths connecting $CU_i$ to $CU_j$. The objective of the optimization problem involves a quadratic objective function that minimizes the least square difference between the wavelength topology and the target traffic matrix. The selection of a quadratic objective function is driven by its convex nature, which facilitates the convergence of the optimization algorithm. This optimization is subject to (1) ingress and (2) egress constraints, which ensures that the number of wavelengths remains within the limits of the available transmitter/receiver wavelengths. Since this optimization considers the globally available bandwidth across all dimensions, the solution to the ILP problem corresponds to the total number of wavelengths per CU-pair. To reduce complexity, we relax the last constraint from an integer to a continuous decision variable and employ a heuristic approach to obtain a solution that closely approximates the optimal solution. The heuristic consists of two steps: proper rounding of the decision variables and an iterative assignment process for CUs that have not been assigned the full set of $w$ wavelengths. In the first step, we round the decision variables to their nearest integer values. Next, we iterate over each CU that requires additional wavelengths, assigning weights based on the difference between the currently allocated wavelengths and the ideal number of wavelengths. The wavelengths are then allocated one-at-a-time in a round-robin fashion based on decreasing weight values. We note that the wavelength assignment at each transceiver port and WSS is not trivial. During each step of this wavelength assignment process, we need to ensure that the ingress and egress wavelength constraints as well as the wavelength multiplexing constraints are satisfied across each CU in the global topology. We continue this iterative process until every CU has been assigned the full set of $w$ transmitting wavelengths. The wavelength routing permutation in each WSS can be solved by constructing a vertex-coloring problem, with individual signals as nodes that are connected based on distinctive properties such as input, output, and wavelength. Algorithms such as DSatur [49] could be used to solve this problem:

$$\min_{\Lambda} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (\lambda_{ij} - t_{ij})^2,$$

$$\text{s.t.} \quad 1) \sum_{j=1}^{N} \lambda_{ij} \le w, \quad \forall\, 0 \le i \le N-1,$$

$$2) \sum_{i=1}^{N} \lambda_{ij} \le w, \quad \forall\, 0 \le j \le N-1,$$

$$3)\ \lambda_{ii} = 0,\ \lambda_{ij} \in \mathbb{Z}, \quad \forall\, i, j \in \{0, \ldots, N-1\}. \qquad \textbf{(1)}$$

To translate the heuristic solution into network control, we can partition the process into two key components: transceiver tuning and switch tuning. This requires a centralized SDN-enabled network controller, which is responsible for coordinating the updates to transceivers and switches. We note that controller with similar functionality has already been implemented commercially [16]. The network controller is also responsible for gathering information about the mapping of jobs onto the topology, as well as the selection of collective algorithms by each job. At each reconfiguration time step $t$, the network controller carries out the optimization heuristic to determine the optimal solution for bandwidth allocation $\lambda_{ij}[t]$. We iterate through all the CU-pair and determine whether the current wavelength allocation scheme is different from the one in the previous time step $\lambda_{ij}[t-1]$. If so, we determine the number of wavelengths that need to be switched and configure both the transceiver and WSS to the target states. Our objective is to minimize modifications to the existing wavelength allocation by assigning wavelengths that have not yet been assigned a destination to the current allocation. It is important to note that we assume the reconfiguration frequency to be on a once-per-workload-arrival basis since the traffic pattern remains relatively stable within each training process as mentioned in Section 2.B. This assumption also aligns with similar observations made in [26,27].

## 4. CO-DESIGNED COLLECTIVE ALGORITHM

The Flex-SiPCO collective algorithm is an extension to the SiPCO collective algorithm [18] and takes job mapping and bandwidth reconfiguration into consideration. It follows a similar design principle as the SiPCO algorithm, leveraging the advantages of the Flex-SiPAC's multi-port property and utilizing all available wavelengths in each dimension at every timestep. An example of the original SiPCO all-reduce algorithm can be found in [19]. We describe the Flex-SiPCO all-reduce as an example, noting that other collective operations such as all-gather and reduce-scatter can be derived from all-reduce, and a similar design principle can be applied to all-to-all communication. We assume that job sizes are an integer multiple of $r$ as a job size smaller or equal to $r$ would reduce this algorithm to the mesh-based algorithm. For each job mapped onto the topology, we first determine the maximum level ($\hat{l}$) of switches $s_{\hat{l}}$, $\hat{l} \in [1, L]$ removing, which will disconnect CUs in this job. This step ensures that we use the appropriate levels of WSS in the subsequent steps. Given the current job's CU placement and the reconfigured topology information, we next identify a list of CUs connected to the same $\hat{l}$ level of switches but are not part of the current job. This step ensures that the algorithm can be applied to any general job mapping. The remaining steps of the algorithm closely follow the original SiPCO algorithm in that the local message is evenly partitioned into $r\hat{l}$ chunks. We proceed by organizing these partitioned chunks into $\hat{l}$ groups, with each group consisting of $r$ chunks. We label the chunks with their group index as $g \in [0, \hat{l}-1]$ and chunk index as $c \in [0, r-1]$. Initially, each CU sends its chunks from group $g = l \bmod \hat{l}$ using link $l$, where $l$ ranges from 0 to $\hat{l}-1$, to a maximum of $r$ directly connected destination CUs sharing the same level of WSS. CUs that are listed as non-communicating CUs are excluded from this process. The chunks originally destined for the excluded CUs are rerouted as transit chunks to the CUs at the same indices as the transmitting CU in the $(l+1) \bmod (\hat{l})$ level, preferably with a reconfigured link to provide higher bandwidth. Subsequently, each CU performs a local reduction operation on all the received chunks (including transit chunks in their respective indices). In the following $v \in [1, \hat{l}]$ steps, we repeat the previous step, but with a rotation of the $\hat{l}$ groups of $r$ chunks through the $\hat{l}$ connected links. This rotation ensures that chunks in group $g = (v+l) \bmod \hat{l}$ are sent through link $l$. The rerouted chunks go through the same rotation (with a constant offset) and reduction with the other rerouted chunks at the corresponding index. In these steps, we leverage the multicasting capability of the WSS to multicast the already reduced chunk in group $g$ to directly connected neighbors within the same job. This process is repeated for $\hat{l}-1$ steps, resulting in each CU possessing $\hat{l}$ fully reduced chunks. In the last step, the $\hat{l}$ chunks in group $g = (\hat{l}+l) \bmod (\hat{l})$ are multicast using links in level $l$ so that each CU now has $r\hat{l}$ chunks from all the CUs in the same job, completing the all-reduce process. We then apply this algorithm to all the jobs.

The complete algorithm is presented in Algorithm 1. The latency cost of this algorithm can be represented by the $\alpha + n\beta$ model where $\alpha$ is the link latency per time step, $\beta$ is the transfer time per byte, and $n$ is the message size on a link per unit step [50]. Under this model, the overall latency is upper bounded by $(\hat{l}+1)\left(\alpha + (r-1)\frac{2n}{r\hat{l}}\beta\right)$ since each link transmits at most $(r-1)\frac{2n}{r\hat{l}}$ bytes during each of the $\hat{l}+1$ steps. The latency term is constant, and the bandwidth term is close to optimal

**Algorithm 1. Flex-SiPCO All-Reduce Algorithm**

```
Input: $r$, $U = \{u_i | i \in [0, P-1]\}$, $\Lambda = [\lambda_{ij}] \in \mathbb{R}^{N \times N}$
1:  for each job $i \in [0, P-1]$ do
2:    $u_i \leftarrow$ Set of CUs in job $i$
3:    $\hat{l} \leftarrow \max\{l | \text{removing } s_l \text{ disconnects } u_i\} + 1$
4:    for each $CU_x \in u_i$, $i \in [0, P-1]$ do
5:        Partition the local message into $r\hat{l}$ evenly sized chunks
6:        $C_g^c \leftarrow$ chunk $c \in [0, r-1]$ in group $g \in [0, \hat{l}-1]$
7:          ▷ Label each $r$ contiguous chunks with index
          $g \in [0, \hat{l}-1]$
8:              ▷ Label each chunk within each group
              $c \in [0, r-1]$
9:        for each $l \in [0, \hat{l}-1]$ do
10:           for each $y \in [0, r-1] | \lambda_{xy}^l > 0$ do
11:               Send $\{C_l^c\}$ using link $l \rightarrow CU_y$, $\forall\, CU_y \in u_i$
12:               Send $\{C_l^c\}$ using link $(l+1)\text{mod}(\hat{l})$, $\forall\, CU_y \notin u_i$
13:   for step $v \in [1, \hat{l}]$ do
14:       for each $CU_x \in u_i$, $i \in [0, P-1]$ do
15:           for each $l \in [0, \hat{l}-1]$ do
16:               $g = (v + l) \bmod \hat{l}$
17:               $rc =$ reduced chunks, $fc =$ transit chunks
18:               Bcast $\{C_g^{rc}\}$ using link $l \rightarrow CU_y$, $\forall CU_y \in u_i$,
                  $\lambda_{xy}^l > 0$
19:               Forward $\{C_g^{fc}\}$ to level $(l+1) \bmod (\hat{l})$
```

as we scale to larger values of $r$. In summary, Flex-SiPCO is a generalization of the previously proposed SiPCO algorithm but takes job mapping and topology reconfiguration into consideration. When the job distribution matches the topology size, the algorithm reduces to the original SiPCO algorithm.

## 5. TESTBED EXPERIMENTS

We conduct two small-scale testbed experiments to demonstrate (1) the feasibility of the wavelength reconfigurable switching primitive and (2) the system-level benefit of wavelength reconfiguration in Flex-SiPAC.

### A. Optical Testbed Experiment

Our first experimental demonstration highlights a hardware implementation for achieving multi-wavelength reconfiguration via a spatial-wavelength selective switch. The ability to selectively route an arbitrary number of wavelengths from any input port to any output port supports flexible bandwidth allocation that facilitates efficient collective communication in the Flex-SiPAC architecture. Our testbed setup, as depicted in Fig. 5(a), employs an array of four continuous-wavelength tunable laser sources (CW-TLSs) spaced 200 GHz apart. The laser wavelengths are modulated with a 16 Gb/s PRBS31 using a linear reference modulator. These modulated signals are then coupled into the $4 \times 4 \times 4\lambda$ spatial wavelength-selective chip [Figs. 5(b) and 5(c)]. As a baseline, the switch is capable of dropping one wavelength per output port, simulating the all-to-all optical connection for each dimension in Flex-SiPAC. The output signals are amplified using an erbium-doped fiber amplifier (EDFA) to compensate for losses. Polarization controllers (PCs) are used to maximize the optical power coupled into the chips, and variable optical attenuators (VOAs) are used to reduce power at the photodetector (PD). Optical spectra measurements are taken at 10 MHz resolution using an optical spectrum analyzer (OSA).

To showcase the wavelength reconfigurability of our system, we thermo-optically tune the switch to two distinct configurations. In the first configuration, all wavelengths are directed to a single output port [Fig. 5(d)]. In the second configuration, the four wavelengths are directed to three output ports, with the second output port receiving two wavelengths [Fig. 5(f)]. Figures 5(e), 5(g), and 5(i) depict the captured optical spectra at each output port for each configuration, respectively. We observe that our channels of interest appear at their respective output ports with a crosstalk suppression of 23 dB. We also observe open eyes in all cases [Fig. 5(j)], illustrating error-free operation at 16 Gb/s. These results exemplify the feasibility of the multi-wavelength reconfiguration primitive required to achieve the Flex-SiPAC architecture in the switching plane. Ongoing research is being conducted to demonstrate the feasibility of reconfiguration also in the transceiver plane as mentioned in Section 3.B.2.

### B. System Testbed Experiment

We then demonstrate the system-level performance of a small-scale Flex-SiPAC ($r = 2$, $L = 2$) architecture using 4 Nvidia Tesla M40 GPUs with RoCEv2 enabled Mellanox ConnectX-4 NICs. The testbed setup is shown in Fig. 6(a). To emulate parallel wavelength transmission, each GPU is configured to have a virtual bridge equipped with two 10 Gb/s SFP+ transceivers sending at two different wavelengths (1550.12 nm and 1556.55 nm). Due to resource limitations, we emulate the transparent switching of WSSs by directly connecting the virtual bridges using fibers. We deployed two TensorFlow MobileNetV2 workloads on the cluster, assigning each job to a pair of sequential GPUs.

In the static SiPAC configuration, the second level links remain idle since there is no inter-job traffic that passes through them. To emulate the wavelength reconfiguration process in Flex-SiPAC, we employ link aggregation groups (LAGs) in the Pica8 switch [Fig. 6(b)] among GPUs mapped to the same job and generate random background traffic to ensure sufficient link utilization to activate the LAGs. We compare the performance of Flex-SiPAC with similarly sized SiPAC (static), EPS-based leaf-spine, and electronic BCube topologies. In the leaf-spine topology, a single spine switch connects to two aggregation switches, and each aggregation switch is linked to two GPUs. The BCube topology has the same connections as the Flex-SiPAC but with electrical switches in each level. We use the NCCL algorithm for collective operations and run the training workload over two epochs with a batch size of 128. The network throughput is monitored using the Ryu SDN OpenFlow monitoring program [Fig. 6(c)]. By redirecting wavelengths from the second level links to the first level, we effectively increase the bandwidth to where the traffic is concentrated. Flex-SiPAC demonstrates a reduction of 20%, 22%, and 23% in job completion time (JCT) compared to static SiPAC, BCube, and leaf-spine, respectively.

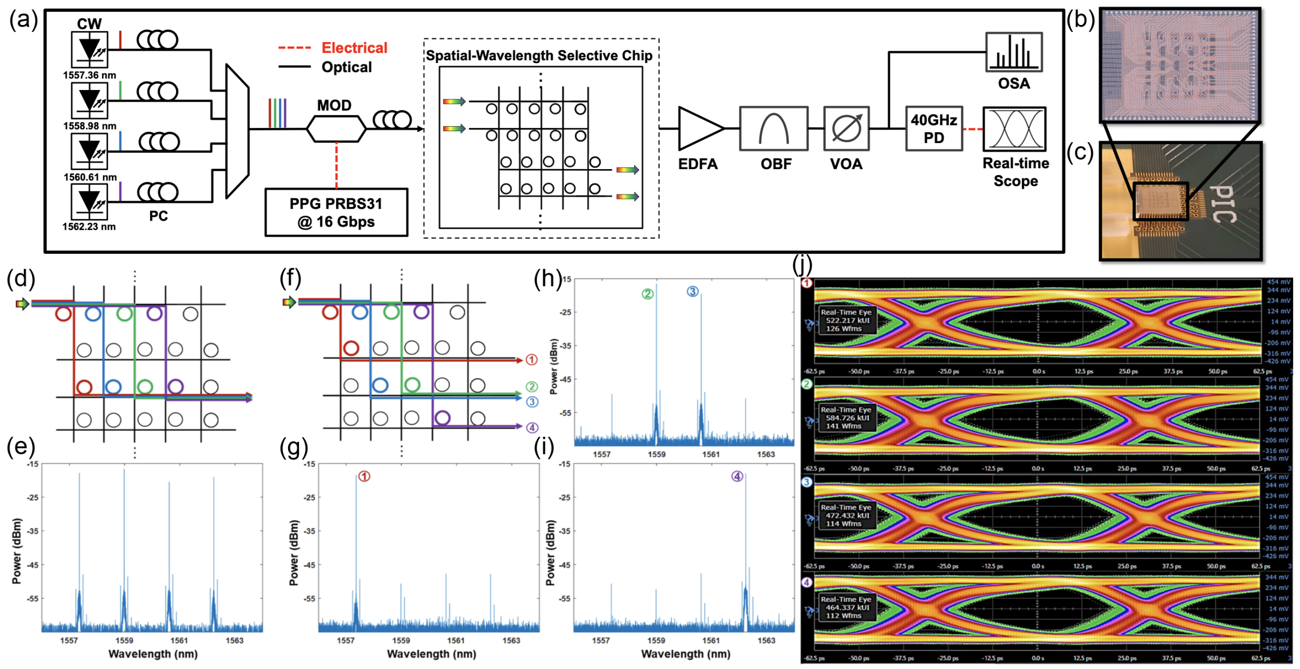

**Fig. 5.** (a) Schematic of the experimental setup. Four CW tunable lasers are used to generate evenly spaced wavelengths at 200 GHz intervals. (b), (c) The signals are modulated with a 16 Gb/s PRBS31 via a linear reference modulator and are coupled into a custom-designed $4 \times 4 \times 4\lambda$ spatial wavelength selective switch. In the first configuration, (d) four wavelengths are routed to the same port, as shown by the optical spectra in (e). And in the second configuration, the four input wavelengths are rerouted to three different ports with the second port getting two wavelengths, as shown by the optical spectra in (g)–(i). The corresponding eye diagrams for each channel in the second configuration can be observed in (j).

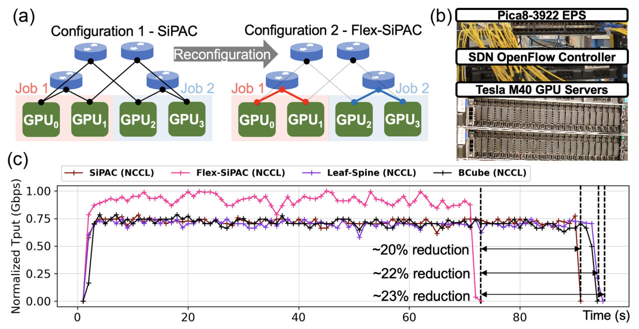

**Fig. 6.** (a) Schematic testbed setup of the SiPAC ($r = 2$, $L = 2$) and the Flex-SiPAC ($r = 2$, $L = 2$) when two jobs of equal sizes are mapped onto the topology. (b) GPU servers connected to an EPS and a SDN controller. (c) Throughput of the injection port under NCCL all-reduce for the static SiPAC, Flex-SiPAC, leaf-spine, and BCube topologies.

## 6. SYSTEM SCALE EVALUATION

### A. Methodology

To demonstrate the scalability of our proposed architecture, we perform extensive packet-level simulations. We use Netbench, an event-driven, packet-level simulator [51], to evaluate the performance of the Flex-SiPAC architecture. We expand the original capabilities of Netbench to incorporate three additional components: (1) support for topologies with diverse link latencies and bandwidths, (2) integration of a link-level back-pressure based loss-less flow control mechanism [27], and (3) handling of traffic with blocking flow starting times (blocking flow starting times refers to the scenario where traffic flows in the current time step cannot start until the traffic flow from the previous time step has concluded, including the completion of the all-reduce operation) commonly found in collective communications. To evaluate the performance of our proposed architecture, the system-scale evaluation comprises two main types of comparisons. We begin by comparing it with state-of-the-art electronic counterparts, followed by a comparison with OCS integrated reconfigurable architectures.

#### 1. *Workloads and Job Mapping*

The primary objective of this study is to assess the architectural performance in the context of heterogeneous multi-tenant job mapping. To accomplish this, we deploy two types of traffic workloads while varying the number of jobs, job sizes, and job mappings to thoroughly test the network. The first type of traffic workload is the primitive *all-reduce* collective, which represents a dominant collective operation in DDL trainings under various parallelisms. The second type of workload comprises realistic *deep learning* traces extracted from application communication task graphs in [30]. The simulated applications include VGG [52], Candle [53], and Transformer (BERT) [3]. For each workload, we simulate iterations of the collective communication, utilizing message sizes extracted from the respective task graphs. Across all architectures, we apply the mesh-based (M) collective algorithm for the all-reduce operations, given their efficiency in multi-port architectures as demonstrated in [19]. As for the Flex-SiPAC architecture, we also employ the Flex-SiPCO all-reduce algorithm (F), as outlined in Section 4. We simulate the relaying of transit chunks by having the transit node initially accept the transit packet and subsequently place it in the output port queue. For each job mapping, we assume that jobs of varying sizes are mapped sequentially onto continuous blocks of CUs in the topology. We represent the job mapping as a set $\{u_i\}$, $i \in [0, P-1]$, where $|u_i|$ represents the number of CUs in job $i$. To quantify the skewness of the job mapping distribution, we utilize the skewness coefficient $\sigma$, defined as

$$\sigma = 1 - \frac{\min(|u_i|)}{\max(|u_i|)}, \quad \forall\, i \in [0, P-1]. \tag{2}$$

A value of $\sigma = 0$ represents the uniform case, where all jobs are of the same size, while $\sigma = 1$ indicates maximum skewness, with jobs varying significantly in size.

### 2. Topologies

We expand upon the list of topologies described in [18,19] by incorporating the latest additions into our evaluation. We normalize the topologies using the per-CU bandwidth.

*H100-SuperPOD* [17]: The basic units of DGX-H100-SuperPOD are DGX-H100 servers, each equipped with eight H100 GPUs connected to an array of four NVSwitches using 18 server-facing NVLinks [54]. Multiple DGX-H100 servers are further interconnected through an additional layer of NVSwitches utilizing 72 network-facing NVLinks per server. This architecture aims to resolve the communication bottleneck caused by bandwidth discrepancies present in earlier SuperPOD iterations [55]. However, its scalability is limited by the physical distance of the NVLinks, restricting the connectivity to a maximum of 256 GPUs. In our study, we adopt the Giant Switch abstraction, assuming that SuperPODs larger than 256 will connect to the same giant switch. We characterize the per-CU bandwidth to be the sum of all NVLinks coming out of a single GPU.

*SiP-OCS*: SiP-OCS is a reconfigurable topology proposed in the SiP-ML DDL training framework [28]. It consists of a layer of $N$ GPUs with radix $Q$ connected to another layer of $Q$ OCSs with radix $N$. For scalability, both the GPUs and the OCSs in this topology require a large radix. In [28], the default number of OCSs is 16. Therefore, we begin with 16 OCSs and scale up to 64 for larger topologies. Initially, the OCSs are configured to distribute bandwidth evenly among as many GPUs as possible. After mapping the jobs onto the topology and acquiring an estimate of the traffic matrix, an ILP formulation that aims to maximize the minimum inverse of the completion time is employed to inform how bandwidth should be steered [28]. In our Netbench simulation pipeline, we implement this estimated traffic matrix assuming a mesh-based all-to-all operation for all jobs. The per-CU bandwidth is defined as the sum of link bandwidths connecting to the OCS layer per GPU.

*TPUv4*: TPUv4 Pod, Google's third supercomputer designed for machine learning workloads, leverages OCSs to dynamically reconfigure its interconnect topology. The basic building blocks of TPUv4 Pod are TPU blocks of size $64 = 4^3$, which are interconnected into a 3D-torus topology using electrical cables. Only the TPUs positioned on the outer surface of these blocks have optical connections to the OCS layer. A 4096 TPU cluster is formed by connecting 64 of these blocks using 48 Palomar OCSs with 136 ports each. The OCSs can be reconfigured to offer varying connection bandwidth among the 64 blocks. In this study, we adopt a straightforward rewiring principle: when a job spans multiple blocks, the OCSs are configured to evenly distribute light paths along all available dimensions among the blocks within the same job. Since we assume sequential job mapping onto the topology, the resulting logical topology takes the form of extended rings along the dimensions that connect these blocks. Here, the per-CU bandwidth is defined as the cumulative bandwidth a single CU has with its neighboring CUs.

## B. Simulation Results

### 1. Primitive All-Reduce Workload

In this experiment, we set the topology size to 512, with a per-CU bandwidth of 1920 Gb/s, equivalent to 60 wavelengths modulated at 32 Gb/s per wavelength. We varied the message size from 1 kB to 100 MB, following common DDL workloads [56–58]. For each topology, we employed a range of job mapping skewness, with the job that spans the whole topology (512 CUs, $\sigma = 1$) serving as the baseline. The final JCT for a multi-tenant job mapping was determined by selecting the maximum JCT among all the jobs in this distribution. As shown in Fig. 7, the Flex-SiPAC topology outperforms the other topologies for small message sizes and exhibits equivalent or better support for larger message sizes before reaching saturation. At the 1 MB data point, prior to the latency curves reaching saturation, the Flex-SiPAC architecture demonstrates a 26% to 29% improvement in JCT compared to the DGX-H100 SuperPOD architecture. This improvement can be attributed to two key factors: the Flex-SiPAC's ability to facilitate simultaneous direct transmissions to and from a large number of endpoints without intermediate switch buffering, and its ability to allocate higher bandwidth to CUs engaged in intensive communication involved the same job. We also note that in the context of a single job ($\sigma = 1$), both the Flex-SiPAC topology and the Flex-SiPCO algorithm converge to the static case. This results in SiPAC and Flex-SiPAC exhibiting identical performance. To maintain figure clarity, we omitted the overlapping line representing the SiPAC-SiPCO combination. Furthermore, Flex-SiPAC is able to achieve comparable network performance with fewer components as it has a reduced component count (transceivers and switches) comparing to the other topologies [19] at given topology sizes.

### 2. Deep Learning Workloads

We next examine the performance of the reconfigurable architectures using realistic DL communication workloads. Figure 8 shows the performance of various topology-collective combinations at 64 CUs [Fig. 8(a)] and 512 CUs [Fig. 8(b)], both with a normalized per-CU bandwidth of 1920 Gb/s. We map *two* jobs, both from the same workload but of varying job sizes, onto the different topologies. The job size distribution varies with a constant decrement in skewness. When examining different workloads, we find that the TPUv4 architecture exhibits worse performance than the other architectures on the VGG workload, especially at higher job mapping skewness. In

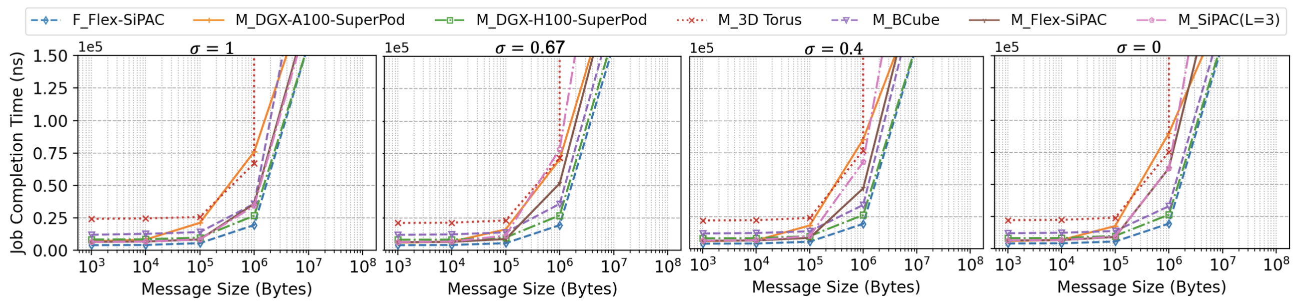

**Fig. 7.** JCT of all-reduce collective communications for 512 CUs across different message sizes and different job mapping skewness ($\sigma$) using mesh-based (M) and Flex-SiPCO (F) all-reduce algorithms.

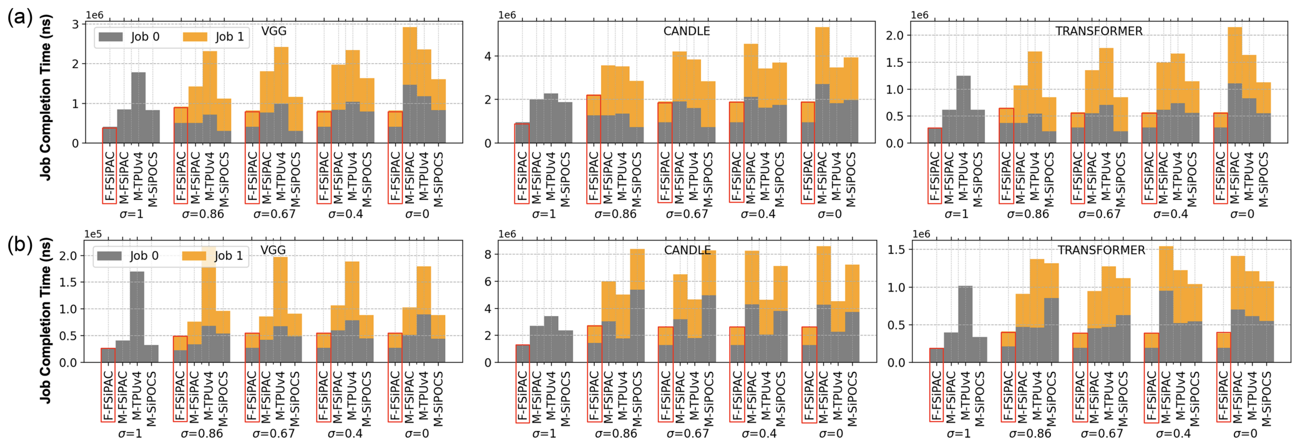

**Fig. 8.** JCT of different topology-algorithm combinations at (a) $N = 64$ and (b) $N = 512$ for three types of DDL workloads at various job mapping skewness ($\sigma$).

scenarios where job mapping distributions are more skewed (with $\sigma$ approaching 1), a larger number of CUs need to communicate with each other. The TPUv4 architecture's 3D-torus construction introduces a large hop count to reach a large number of destinations, resulting in increased latency costs and intermediate CU buffering. This effect becomes particularly apparent when considering the VGG workload, which consists of relatively smaller message sizes that are more sensitive to latency. However, the reconfigured Flex-SiPAC architecture demonstrates distinct advantages in this scenario. Its multi-wavelength simultaneous transmission capability enables efficient communication to a larger number of destinations with minimal hop count, effectively reducing the latency cost associated with mesh-based operations. Conversely, when jobs are uniformly mapped (with $\sigma$ close to 0), a significant portion of links remain uninvolved in communication. In the worst-case scenario, half of the CUs may not communicate with the other half, resulting in idle dedicated direct-connect links. Despite efforts to reconfigure the wavelengths, the redirected bandwidth remains limited to links connecting CUs within the same job to their directly connected neighbors. For mesh-based algorithms that require simultaneous transmission to all CUs within a job, this necessitates multi-hop transmission and intermediate CU buffering, leading to congestion. In this context, the combination of the mesh algorithm with Flex-SiPAC demonstrates worse performance. This characteristic becomes evident when analyzing the results for the candle and transformer workloads, which feature relatively larger per-iteration message sizes. However, Flex-SiPCO effectively addresses this concern by enabling transmissions exclusively to directly connected neighbors. Although this introduces slightly larger per time-step message chunks, it results in reduced overall congestion and queuing delays, striking a beneficial trade-off. In summary, we observe that the Flex-SiPAC-Flex-SiPCO co-design is able to outperform the other topology-collective combinations across different topology sizes and job mapping skewness.

**Scalability Analysis:** We assess the scalability of the Flex-SiPAC architecture by comparing it to the other switch-based architectures, normalizing the results based on their switch radix, since it is a key factor when determining the maximum attainable scale of a reconfigurable network. For DGX-SuperPOD, we use the radix of the top-layer NVSwitch and assume a constant per-server aggregation of eight GPUs. From Fig. 9, we observe that SiP-OCS exhibits poor scalability as its current architecture assumes that each OCS has the same number of ports as the CUs in the flat topology. The DGX-H100 scales slightly better than SiP-OCS as it has an extra layer of server aggregation. TPUv4 demonstrates better scalability due to its topology construction: the building blocks of 64 TPUs are fixed, and the OCS radix limits the number of wrap-around links that can be connected, resulting in higher aggregation under the OCS. A 2-level Flex-SiPAC exhibits

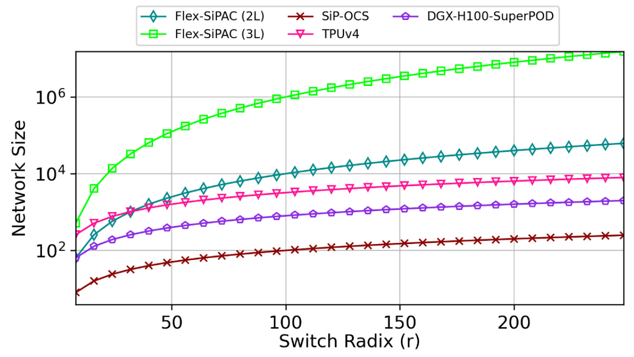

**Fig. 9.** Network size as a function of switch radix for the reconfigurable accelerator network topologies.

improved scalability compared to TPUv4 after $r = 32$ as the number of endpoints scale exponentially with base $r$. And lastly, the 3-level Flex-SiPAC scales more effectively due to the exponential increase in endpoints with the number of levels. A 3-level Flex-SiPAC using WSS radix of 16 would give us 4096 endpoint CUs (the same size as the production TPUv4 cluster) and is therefore enough to construct an accelerator cluster in a data center network with a reasonable switch radix and a small network diameter. The bisection bandwidth of the Flex-SiPAC architecture scales proportionally with the number of endpoints because all CUs have one connection to the top-layer WSS in the other half of the topology.

## 7. CONCLUSION

In this study, we describe the Flex-SiPAC accelerator network architecture designed to facilitate efficient collective communication in DDL. By leveraging wavelength reconfigurable transceivers and WSSs, Flex-SiPAC achieves a highly flexible multi-dimensional all-to-all network. Our experimental testbed demonstrates the WSS's ability to achieve compact and arbitrary bandwidth steering, thereby validating the feasibility of wavelength reconfiguration in the Flex-SiPAC architecture. In addition to the physical hardware, we also proposed an all-reduce collective algorithm that would efficiently use the bandwidth in the Flex-SiPAC architecture. To further evaluate Flex-SiPAC's performance, we conduct realistic packet-level simulations, considering factors such as topology size, message size, and job mapping skewness. The results of our system-level simulations demonstrate that the topology-algorithm co-design is able to achieve a 26% to 29% reduction in communication time compared to current state-of-the-art compute clusters in representative collective communications.

**Funding.** Advanced Research Projects Agency - Energy (DE-AR00000843); National Security Agency (FA8075-14-D-0002-0007, TAT 15-1158); Center for Ubiquitous Connectivity (CUbiC), sponsored by the Semiconductor Research Corporation (SRC) and the Defense Advanced Research Projects Agency (DARPA) under the JUMP 2.0 program.

**Disclosures.** The authors declare no conflicts of interest.

## REFERENCES

1. D. Narayanan, M. Shoeybi, J. Casper, *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).
2. A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017).
3. J. Devlin, M.-W. Chang, K. Lee, *et al.*, "BERT: pre-training of deep bidirectional transformers for language understanding," arXiv, arXiv:1810.04805 (2018).
4. "Nvidia clocks world's fastest BERT training time and largest transformer based model, paving path for advanced conversational AI" (2019), https://developer.nvidia.com/blog/training-bert-with-gpus/.
5. Z. Yang, Z. Dai, Y. Yang, *et al.*, "XLNet: generalized autoregressive pretraining for language understanding," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (2019), pp. 5753–5763.
6. D. Lepikhin, H. Lee, Y. Xu, *et al.*, "GShard: scaling giant models with conditional computation and automatic sharding," arXiv, arXiv:2006.16668 (2020).
7. D. Patterson, J. Gonzalez, Q. Le, *et al.*, "Carbon emissions and large neural network training," arXiv, arXiv:2104.10350 (2021).
8. S. Smith, M. Patwary, B. Norick, *et al.*, "Using DeepSpeed and Megatron to train Megatron-Turing NLG 530B, a large-scale generative language model," arXiv, arXiv:2201.11990 (2022).
9. D. Mudigere, Y. Hao, J. Huang, *et al.*, "Software-hardware co-design for fast and scalable training of deep learning recommendation models," in *Proceedings of the 49th Annual International Symposium on Computer Architecture* (2022), pp. 993–1011.
10. A. Chowdhery, S. Narang, J. Devlin, *et al.*, "PaLM: scaling language modeling with pathways," arXiv, arXiv:2204.02311 (2022).
11. H. Touvron, T. Lavril, G. Izacard, *et al.*, "LLaMA: open and efficient foundation language models," arXiv, arXiv:2302.13971 (2023).
12. R. Thoppilan, D. De Freitas, J. Hall, *et al.*, "LaMDA: language models for dialog applications," arXiv, arXiv:2201.08239 (2022).
13. T. L. Scao, A. Fan, C. Akiki, *et al.*, "BLOOM: a 176B-parameter open-access multilingual language model," arXiv, arXiv:2211.05100 (2022).
14. "TrendForce says with cloud companies initiating AI arms race, GPU demand from ChatGPT could reach 30,000 chips as it readies for commercialization" (2023), https://www.trendforce.com/presscenter/news/20230301-11584.html.
15. M. Hamblen, "Update: ChatGPT runs 10K Nvidia training GPUs with potential for thousands more" (2023), https://www.fierceelectronics.com/sensors/chatgpt-runs-10k-nvidia-training-gpus-potential-thousands-more.
16. N. P. Jouppi, G. Kurian, S. Li, *et al.*, "TPU v4: an optically reconfigurable supercomputer for machine learning with hardware support for embeddings," arXiv, arXiv:2304.01433 (2023).
17. "NVIDIA H100 tensor core GPU architecture" (2023), https://resources.nvidia.com/en-us-tensor-core.
18. Z. Wu, L. Y. Dai, Z. Zhu, *et al.*, "SiP architecture for accelerating collective communication in distributed deep learning," in *Optical Fiber Communication Conference (OFC)* (2023), paper W1G.1.
19. Z. Wu, L. Y. Dai, A. Novick, *et al.*, "Peta-scale embedded photonics architecture for distributed deep learning applications," J. Lightwave Technol. **41**, 3737–3749 (2023).
20. "Google cloud TPU" (2024), https://cloud.google.com/tpu/docs/intro-to-tpu.
21. K.-I. Sato, H. Matsuura, R. Konoike, *et al.*, "Prospects and challenges of optical switching technologies for intra data center networks," J. Opt. Commun. Netw. **14**, 903–915 (2022).
22. A. Roy, H. Zeng, J. Bagga, *et al.*, "Inside the social network's (datacenter) network," in *Proceedings of the Special Interest Group on Data Communication* (2015), pp. 123–137.
23. X. Guo, F. Yan, J. Wang, *et al.*, "RDON: a rack-scale disaggregated data center network based on a distributed fast optical switch," J. Opt. Commun. Netw. **12**, 251–263 (2020).
24. J. L. Benjamin, T. Gerard, D. Lavery, *et al.*, "PULSE: optical circuit switched data center architecture operating at nanosecond timescales," J. Lightwave Technol. **38**, 4906–4921 (2020).
25. M. Y. Teh, S. Zhao, P. Cao, *et al.*, "Enabling quasi-static reconfigurable networks with robust topology engineering," IEEE/ACM Trans. Netw. **31**, 1056–1070 (2022).
26. L. Poutievski, O. Mashayekhi, J. Ong, *et al.*, "Jupiter evolving: transforming google's datacenter network via optical circuit switches

and software-defined networking," in *Proceedings of the ACM SIGCOMM 2022 Conference* (2022), pp. 66–85.
27. M. Y. Teh, Z. Wu, M. Glick, *et al.*, "Performance trade-offs in reconfigurable networks for HPC," J. Opt. Commun. Netw. **14**, 454–468 (2022).
28. M. Khani, M. Ghobadi, M. Alizadeh, *et al.*, "SiP-ML: high-bandwidth optical network interconnects for machine learning training," in *Proceedings of the ACM SIGCOMM 2021 Conference* (2021), pp. 657–675.
29. Y. Lu, H. Gu, X. Yu, *et al.*, "X-NEST: a scalable, flexible, and high-performance network architecture for distributed machine learning," J. Lightwave Technol. **39**, 4247–4254 (2021).
30. W. Wang, M. Khazraee, Z. Zhong, *et al.*, "TopoOpt: co-optimizing network topology and parallelization strategy for distributed training jobs," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)* (2023).
31. M. Fariborz, X. Xiao, P. Fotouhi, *et al.*, "Silicon photonic Flex-LIONS for reconfigurable multi-GPU systems," J. Lightwave Technol. **39**, 1212–1220 (2021).
32. M. Shoeybi, M. Patwary, R. Puri, *et al.*, "Megatron-LM: training multi-billion parameter language models using model parallelism," arXiv, arXiv:1909.08053 (2019).
33. A. Shah, V. Chidambaram, M. Cowan, *et al.*, "TACCL: guiding collective algorithm synthesis using communication sketches," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)* (2023), pp. 593–612.
34. A. Gibiansky, "Bringing HPC techniques to deep learning," Baidu Research Tech. Rep. (2017).
35. X. Jia, S. Song, W. He, *et al.*, "Highly scalable deep learning training system with mixed-precision: training ImageNet in four minutes," arXiv, arXiv:1807.11205 (2018).
36. S. Wang, J. Geng, and D. Li, "Impact of synchronization topology on DML performance: both logical topology and physical topology," IEEE/ACM Trans. Netw. **30**, 572–585 (2021).
37. C. Guo, G. Lu, D. Li, *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.
38. B. Corcoran, M. Tan, X. Xu, *et al.*, "Ultra-dense optical data transmission over standard fibre with a single chip source," Nat. Commun. **11**, 2568 (2020).
39. J. Sun, R. Kumar, M. Sakib, *et al.*, "A 128 Gb/s PAM4 silicon microring modulator with integrated thermo-optic resonance tuning," J. Lightwave Technol. **37**, 110–115 (2018).
40. Q. Cheng, M. Bahadori, Y.-H. Hung, *et al.*, "Scalable microring-based silicon CLOS switch fabric with switch-and-select stages," IEEE J. Sel. Top. Quantum Electron. **25**, 3600111 (2019).
41. K. Wen, P. Samadi, S. Rumley, *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.
42. M. Y. Teh, Z. Wu, and K. Bergman, "Flexspander: augmenting expander networks in high-performance systems with optical bandwidth steering," J. Opt. Commun. Netw. **12**, B44–B54 (2020).
43. G. Liu, R. Proietti, M. Fariborz, *et al.*, "Architecture and performance studies of 3D-Hyper-Flex-LION for reconfigurable all-to-all HPC networks," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* (IEEE, 2020), paper 26.
44. L. Y. Dai, V. Gopal, and K. Bergman, "Ultra-scalable microring-based architecture for spatial-and-wavelength selective switching," in *IEEE 19th International Conference on Group IV Photonics (GFP)* (IEEE, 2023), paper WC3.
45. Y. Wang, A. Novick, R. Parsons, *et al.*, "Scalable architecture for sub-pJ/b multi-Tbps comb-driven DWDM silicon photonic transceiver," Proc. SPIE **12429**, 271–288 (2023).
46. A. Rizzo, A. Novick, V. Gopal, *et al.*, "Integrated Kerr frequency comb-driven silicon photonic transmitter," arXiv, arXiv:2109.10297 (2021).
47. S. Wang, A. Novick, A. Rizzo, *et al.*, "Integrated, compact, and tunable band-interleaving of a Kerr comb source," in *CLEO: Science and Innovations* (Optica Publishing Group, 2023), paper STh3J.6.
48. A. Rizzo, S. Daudlin, A. Novick, *et al.*, "Petabit-scale silicon photonic interconnects with integrated Kerr frequency combs," IEEE J. Sel. Top. Quantum Electron. **29**, 3700120 (2023).
49. D. Brélaz, "New methods to color the vertices of a graph," Commun. ACM **22**, 251–256 (1979).
50. R. Thakur, R. Rabenseifner, and W. Gropp, "Optimization of collective communication operations in MPICH," Int. J. High Performance Comput. Appl. **19**, 49–66 (2005).
51. Netbench, https://github.com/ndal-eth/netbench.
52. J. M. Wozniak, H. Yoo, J. Mohd-Yusof, *et al.*, "High-bypass learning: automated detection of tumor cells that significantly impact drug response," in *IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)* (IEEE, 2020).
53. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv, arXiv:1409.1556 (2014).
54. "Upgrading multi-GPU interconnectivity with the third-generation NVIDIA NVSwitch" (2022), https://developer.nvidia.com/blog/upgrading-multi-gpu-interconnectivity-with-the-third-generation-nvidia-nvswitch/.
55. "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.
56. K. Tanaka, Y. Arikawa, K. Kawai, *et al.*, "Large-message size allreduce at wire speed for distributed deep learning," poster session presented at SC18: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (2018).
57. M. Naumov, J. Kim, D. Mudigere, *et al.*, "Deep learning training in Facebook data centers: design of scale-up and scale-out systems," arXiv, arXiv:2003.09518 (2020).
58. J. Fei, C.-Y. Ho, A. N. Sahu, *et al.*, "Efficient sparse collective communication and its application to accelerate distributed deep learning," in *Proceedings of the ACM SIGCOMM 2021 Conference* (2021), pp. 676–691.