2023 32nd International Conference on Parallel Architectures and Compilation Techniques (PACT) | 979-8-3503-4254-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/PACT58117.2023.00028

A Silicon Photonic Multi-DNN Accelerator

Yuan Li*, Ahmed Louri*, Avinash Karanth†
*Department of Electrical and Computer Engineering, George Washington University, Washington, DC, USA
†School of Electrical Engineering and Computer Science, Ohio University, Athens, Ohio, USA
Emails: *{liyuan5859, louri}@gwu.edu, †karanth@ohio.edu

***Abstract*—In shared environments like cloud-based datacenters, hardware accelerators are deployed to meet the scale-out computation demands of deep neural network (DNN) inference tasks. As conventional hardware accelerators optimized for single-DNN execution cannot effectively resolve the dynamic interaction of these inference-as-a-service (INFaaS) tasks, several multi-DNN hardware accelerators have been developed to improve the overall system performance while adhering to the constraints of individual tasks. Some of such multi-DNN hardware accelerators temporally schedule tasks by incorporating the preemption or load-balancing-based algorithm but suffer from resource underutilization because of unmanaged mismatch between resource demand and provision. Other multi-DNN hardware accelerators enable spatial colocation of tasks to improve resource utilization and system flexibility, but the irregular communication patterns between the fragmented resource partitions cannot be adequately supported by the metallic-based interconnects due to their rigidity and other inherent scaling limitations. We introduce a photonic multi-DNN accelerator named ASPIRE in this paper. The fundamental novelty of ASPIRE lies in the ability to adaptively create sub-accelerators for different tasks by assembling fine-grained resource partitions in the same architecture. Seamless communications between those fragmented resource partitions from a sub-accelerator are realized by exploiting photonic interconnects. Specifically, ASPIRE includes three novel designs: (1) a photonic network that can be adaptively partitioned into several sub-networks, each seamlessly connecting the fragmented resource partitions to construct sub-accelerators; (2) a dataflow that simultaneously leverages temporal and spatial data reuse opportunities within each resource partition and across several resource partitions, respectively; (3) an algorithm that allocates resource partitions at task granularity and derives optimal tile size and execution order at DNN layer granularity. Simulation studies show that ASPIRE outperforms other state-of-the-art multi-DNN accelerators, delivering 64% execution time reduction, 69% energy saving, 51% improvement in service-level agreement satisfaction rate, and 7.9× improvement in fairness.**

***Index Terms*—Deep neural network, Accelerator, Silicon photonics**

I. INTRODUCTION

Large-scale accelerators are increasingly being deployed in shared multi-DNN environments (such as in cloud data centers [1]–[4]) in order to meet the demands of large-scale compute-intensive deep neural network (DNN) workloads. Typically, these inference-as-a-service (INFaaS) requests from different DNN applications are satisfied by partitioning the large accelerator into multiple smaller accelerators by distributing the workloads and allocating resources to each inference request [5]–[8]. As INFaaS demands increase with stringent quality of service (QoS) guarantees for DNN applications, DNN accelerators will be required to allocate resources incrementally while allowing seamless communication for data movement. Most prior single-task execution-based DNN accelerators [1], [9]–[19] cannot be directly utilized for multi-DNN workload since the underlying hardware was not designed for guaranteeing fairness or other service-level agreements (SLA). Further, naively applying single-task DNN accelerators to multi-DNN workloads can also lead to underutilized hardware resources which can impact throughput and increase latency.

Several temporal multi-DNN accelerators [5], [6] have been proposed in the literature to address the QoS and fairness problems by incorporating preemption [5] and load-balancing [6] into task scheduling. This allows resources such as PE arrays to be allocated while load-balancing the workload with QoS guarantees. Due to temporal variations in INFaaS requests, there is a mismatch between computation demand and resources allocated which leads to underutilized resources. For example, systolic-array-based temporal multi-DNN accelerators such as PREMA [5] and AI-MT [6] allocate all processing elements (PEs) to a single task at any time regardless of computation demand which can lead to underutilized PE arrays. On the other hand, recent spatial multi-DNN accelerators such as [7], [8] address the resource underutilization problem by partitioning and allocating only the right amount of hardware resources to each ongoing task, but are often curtailed by the amount of communication between partitioned regions due to the high-latency and power consumption of metallic-based interconnects. For example, systolic-array-based spatial multi-DNN accelerators such as Dataflow Mirroring [8] and Planaria [7] partition the systolic array into several partitions and allocate one or several partitions to each ongoing task. Dataflow Mirroring prohibits communications between partitions and, as a result, lacks flexibility and does not fully exploit spatial parallelism. By contrast, Planaria adopts metallic-based ring-buses for communications between partitions. Although the proposed design is flexible, the communication latency can be significantly affected by the physical location of the partitions.

While temporal and spatial multi-DNN accelerators improve the design, traditional metallic interconnects impose fundamental limits on scale-out performance due to high performance/Watt, complex data movement between distant partitions, and rigidity in allocating compute resources. Traditional metallic interconnects will be required to carry the additional burden of connecting together distant islands of accelerators (PE arrays) for INFaaS requests which can result in higher power consumption than single-task accelerators due to excessive data movement. The data movement problem will exacerbate in the future scale-out accelerators that need to accommodate several INFaaS requests simultaneously.

Silicon photonics is a disruptive technology that has the potential to alleviate the communication issues that limit the performance of traditional electronic interconnects [17], [20]–[30]. Photonic interconnects have several desirable properties such as the (1) ability to design flexible topologies by implementing broadcast, multicast, and unicast, (2) reducing energy consumption by consuming power only at the endpoints of the communication channels, and (3) providing very high bandwidth-density for tighter integration of photonic components. Silicon photonics offers distance-independent latency for connecting distant PE arrays, thereby providing opportunities to seamlessly interconnect PE arrays without any rigidity.

In this paper, we propose a multi-DNN accelerator named **ASPIRE** which exploits both temporal and spatial techniques along with the high flexibility and distance-independent latency properties of silicon photonics. In ASPIRE, we exploit the distance-independent latency feature of silicon photonics to construct sub-accelerators by providing the required computational resources to each of the concurrent inference tasks for the simultaneous transmission of inputs and kernels, even when the arbitrarily partitioned sub-arrays could be physically distant from each other. By interconnecting the sub-array of PEs with photonic waveguides and multiple wavelengths, we can seamlessly interconnect distant PEs, thereby building islands of PEs for different inference tasks. ASPIRE leverages the complete broadcast and multicast capabilities of silicon photonics to design flexible dataflow, thereby improving the performance/Watt. Finally, we propose an algorithm that generates balanced resource allocation decisions, taking both computation demand and task priority into consideration. The algorithm ensures that we can alleviate the resource underutilization problem associated with multi-DNN accelerators. The combined effects of a reconfigurable photonic interconnect that provides broadcast and multicast capabilities along with a task allocation algorithm that ensures fairness and provides QoS guarantees makes ASPIRE a novel multi-DNN accelerator for future DNN workloads. Simulation studies using a collection of diverse DNN models [31]–[37] show that ASPIRE architecture outperforms other state-of-the-art temporal and spatial multi-DNN accelerators [5]–[8] by delivering 61% execution time reduction, 69% energy saving, 51% improvement in service-level agreement satisfaction rate, and 7.9× improvement in fairness. The major contributions of this work are as follows:

- **Technology-Specific Dataflow**: We develop a dataflow that converts a sufficient fraction of data reuse opportunities into data multicast opportunities through spatial parallelism to leverage the energy-efficient multicast property of silicon photonics while not undermining the rest of data reuse opportunities, which can still be leveraged through conventional dataflow optimizations.
- **Reconfigurable Photonic Network**: We design a photonic network that can be adaptively divided into several sub-networks, each seamlessly connecting some fragmented PE partitions to construct a sub-accelerator for a task. Silicon photonic interconnects facilitate flexible network reconfiguration and low-overhead communication between distant partitions.
- **Allocation Algorithm**: We propose an allocation algorithm that produces a balanced allocation of PE partitions based on computing demand and task priority, and allocation of on-chip memory capacity based on DNN topology and data reuse pattern, while conventional allocation algorithms focus on dividing computing resource only.

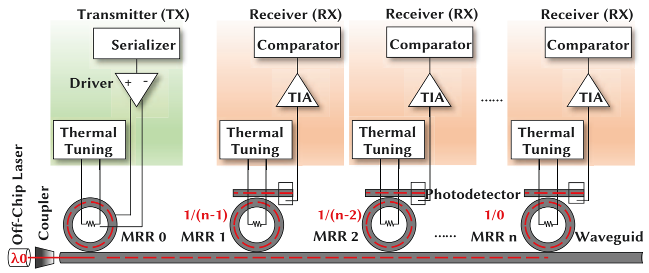

Fig. 1. A single-write-multiple-read (SWMR) silicon photonic channel using wavelength $\lambda0$ with one transmitter and n receivers, each equipped with a microring resonator (MRR).

II. Background and Motivation

A. Photonic Interconnects

Figure 1 shows a photonic interconnect with one transmitter and n receivers connected by a waveguide. The light including wavelength $\lambda0$ is generated by the off-chip laser source and coupled to the waveguide through an optical coupler [38]. Within the transmitter, a microring resonator (MRR) [39] labeled MRR 0 works as an optical modulator to modulate on $\lambda0$ with input data as modulation signals. Within the n receivers, MRRs labeled MRR 1-n work as optical filters to forward a fraction of light in $\lambda0$ to the local photodetectors [20] and the rest of light in $\lambda0$ to downstream receivers. The electrical signals generated by the local photodetectors are amplified by the transimpedance amplifiers (TIAs) and sent to comparators to retrieve the original data. Each MRR works as either an optical modulator or optical filter and is tuned by a resistive heater controlled by a thermal tuning unit to mitigate thermal and process variations [20]. Please note that the MRR in each receiver is tuned to have a specific split ratio based on the number of downstream receivers. For example, the MRR in the first receiver is tuned to have a split ratio of 1/(n-1) as there are $n-1$ downstream receivers. The MRR in the last receiver is tuned to have a split ratio of 1/0, which is equivalent to the on-resonant state. Since all receivers as shown in Figure 1 receive the same photonic signals which are then retrieved to the same data, the photonic interconnect in Figure 1 acts as a single-write-multiple-read (SWMR) channel, which is especially energy-efficient for multicast communication. An SWMR channel achieves 0.1 $pJ/bit/receiver$ communication (using parameters listed in Section 6.1), which is 17× lower than the metallic-based wires with the state-of-the-art ground-referenced signaling (GRS) technique [9]. This work leverages the energy-efficient multicast property of silicon photonics and

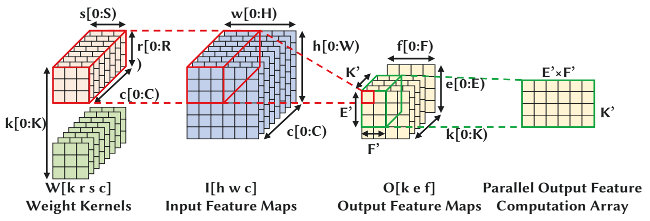

Fig. 2. Computations in a convolutional layer $\langle K\ E\ F\ C\ R\ S\rangle$.

Algorithm 1: Computation of a Convolutional Layer

```
1 for ( $k = 0$; $k < K$; $k$ += 1 )
2   for ( $e = 0$; $e < E$; $e$ += 1 )
3     for ( $f = 0$; $f < F$; $f$ += 1 )
4       for ( $c = 0$; $c < C$; $c$ += 1 )
5         for ( $r = 0$; $r < R$; $r$ += 1 )
6           for ( $s = 0$; $s < S$; $s$ += 1 )
7             $O[k\ e\ f]$ += $I[r+e-1\ s+f-1\ c] \times W[k\ r\ s\ c]$
```

investigates the optimization opportunities of promoting data multicast opportunities through spatial parallelism.

B. DNN Communication

The computations involved in a typical convolutional layer in DNN inference tasks shown in Figure 2 can be represented by a nested loop over weight kernels, input feature maps (ifmaps), and output feature maps (ofmaps) as shown in Algorithm 1. The nested loop includes iterations on six dimensions: the number of ofmaps $\langle k\rangle$, the height $\langle e\rangle$ and width $\langle f\rangle$ of ofmaps, the number of ifmaps $\langle c\rangle$, and the height $\langle r\rangle$ and width $\langle s\rangle$ of weight kernels. Please note that the height $\langle h\rangle$ and width $\langle w\rangle$ of ifmaps are not independent dimensions and can be represented by the above dimensions. Figure 2 shows the computations of an iteration in the $\langle f\rangle$ loop ($Line$ 3 of Algorithm 1) to generate a specific output $O[k\,e\,f]$. Since DNN hardware accelerators often leverage parallelism among a large amount of PEs, we can generate $K^{'} \times \left(E^{'} \times F^{'}\right)$ outputs in parallel using a PE array with height and width equal $K^{'}$ and $E^{'} \times F^{'}$, respectively (processing all or part of $Line$ 1-3 of Algorithm 1 in parallel). This PE array is considered the sub-accelerator for a given inference task and can be constructed by combining several PE partitions in the system.

Data communication when processing a convolutional layer includes transmitting weight kernels and ifmaps to PEs and collecting the generated ofmaps or partial sums (psums) from PEs. Unlike generic applications, data communication for DNN inference tasks is regular and can be derived from parameters in the nested loop and underlying hardware accelerator. From Figure 2, we can observe four types of data reuse opportunities: (1) weight data reuse opportunities due to convolutional operations in $\langle e\ f\rangle$ dimensions; (2) input data reuse opportunities due to convolutional operations in $\langle e\ f\ r\ s\rangle$ dimensions; (3) input data reuse opportunities due to multiple weight filter channels in $\langle k\rangle$ dimension; (4) psum reuse opportunities due to local accumulation in $\langle c\ r\ s\rangle$ dimensions. Please note that data reuse opportunities in categories (1) and (2) cannot be simultaneously exploited due to the pair-wise multiplication nature of convolutional operations. Spatial parallelism can convert data reuse opportunities into data multicast opportunities. For example, spatial parallelism in the $\langle k\rangle$ dimension completely converts data reuse opportunities in category (3) into data multicast opportunities while data reuse opportunities in categories (1), (3), and (4) are not undermined. By contrast, spatial parallelism in the $\langle c\rangle$ dimension does not generate any data multicast opportunities while partially undermining data reuse opportunities in category (4). The target of this work is to convert a sufficient amount of data reuse opportunities into data multicast opportunities to leverage the energy-efficient multicast property of silicon photonics while not undermining the rest of the data reuse opportunities, which can still be leveraged through conventional dataflow optimizations.

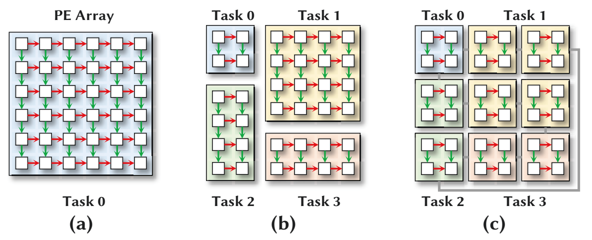

Fig. 3. (a) Systolic array based accelerator with metallic-based interconnects between adjacent PEs [1]. (b) Spatial multi-DNN accelerator with 4 partitions. (c) Spatial multi-DNN accelerator with 4 sub-accelerators constructed by smaller partitions and ring bus network.

C. Inter-Partition Communication Bottleneck

Spatial multi-DNN accelerators require partitioning the uniform PE array into smaller partitions so that a single partition or a collection of partitions can be allocated to a particular inference task [7], [8]. The inter-partition communication is often considered the system bottleneck. Consider, for example, the systolic-array-based accelerator [1], which is the baseline architecture, of many prior multi-DNN accelerators [5]–[8] as shown in Figure 3 (a). PEs are organized in a planar array and augmented with distributed buffers for weights and inputs. Metallic-based interconnects are implemented between adjacent PEs to facilitate inter-PE data reuse. When partitioning the uniform PE array into smaller partitions for concurrent inference tasks, the metallic-based interconnects on partition boundaries are removed. In this case, the inter-partition communication fabric determines how partitions can be combined and allocated to concurrent inference tasks.

As shown in Figure 3(b), Dataflow Mirroring [8] partitions the uniform 6×6 PE array into four partitions of different sizes and allocates each partition to a given inference task. The intra-partition communication is supported by the metallic-based interconnects in the original systolic array while the inter-partition communication is not supported. Since Dataflow Mirroring enforces the constraint that each inference task must be allocated with a physical PE array, it can only accommodate up to four concurrent inference tasks, which does not fully exploit the spatial task parallelism on the accelerator. By contrast, as shown in Figure 3(c), Planaria [7] partitions the uniform 6×6 PE array into identical 2×2 partitions and allocates a partition or a collection of partitions to a given inference task. Several fragmented partitions may act as a sub-accelerator and communicate through ring buses [7]. For inference $Task$ 1 shown in Figure 3(c), the communication between the partitions takes one to four hops. The number of communication hops can be even higher when partition size decreases, system scales, or when communicating partitions are distant from each other, which negatively affects the performance and energy efficiency of the multi-DNN accelerator.

In addition to the common advantages over metallic-based interconnects such as high communication bandwidth density and high energy efficiency [20]–[22], [26], the distance-independent latency feature [20] of silicon photonics can potentially tackle the inter-partition communication challenge and enable the construction of the sub-accelerator for a particular inference task by combining fragmented partitions via seamless inter-partition communication support. For example, by adopting photonic interconnects instead of the metallic-based ring buses shown in Figure 3(c), one-hop communication can be achieved between any partitions allocated to inference $Task$ 1. We implement only photonic interconnects in the ASPIRE architecture so that one-hop communication is achieved between fragmented partitions and a global buffer (GLB) [40].

III. ASPIRE DATAFLOW

Section 2.2 suggests that data reuse opportunities can be turned into data multicast opportunities when performing operations in certain dimensions in parallel. Here we examine the impact of parallelism in different dimensions. Parallelism in the $\langle k \rangle$ dimension completely turns input data reuse opportunities due to multiple weight filters into multicast opportunities while maintaining all other data reuse opportunities. Parallelism in the $\langle e\, f \rangle$ dimensions turns either weight or input data reuse opportunities due to convolutional operations into multicast opportunities while maintaining all other data reuse opportunities. Parallelism in the $\langle c \rangle$ dimension does not create any multicast opportunities while partially undermining psum data reuse opportunities. Parallelism in the $\langle r\, s \rangle$ dimensions turns input data reuse opportunities due to convolutional operations into multicast opportunities while partially undermining psum data reuse opportunities. Since the optimization target of the ASPIRE dataflow is to promote data multicast opportunities while maintaining data reuse opportunities, we decide to perform operations in $\langle k\, e\, f \rangle$ dimensions in parallel. Algorithm 2 is one possible configuration of the proposed ASPIRE dataflow with parallelism in $\langle k\, e\, f \rangle$ dimensions and an emphasis on psum data reuse. P_k, P_e, and P_f represent the number of PEs working in parallel in the $\langle k \rangle$, $\langle e \rangle$, and $\langle f \rangle$ dimensions, respectively.

Please note that the proposed ASPIRE dataflow has a variety of configurations. We can generate a configuration with an emphasis on weight data reuse by moving the loops in $\langle c\, r\, s \rangle$ dimensions before the loop in the $\langle e \rangle$ dimension, in other words, moving $Line$ 7-9 before $Line$ 2 in Algorithm 2. We can also generate a configuration with an emphasis on input data reuse by moving the loops in $\langle c\, r\, s \rangle$ dimensions after the loop in the $\langle f \rangle$ dimension and moving the loop in the $\langle k \rangle$ dimension after the loop in the $\langle c \rangle$ dimension, in other words, moving $Line$ 7-9 after $Line$ 3 and moving $Line$ 1 before $Line$ 4 in Algorithm 2. The other ASPIRE dataflow configurations can be derived by changing the order of the six loops ($Line$ 1-3 and $Line$ 7-9 in Algorithm 2).

Algorithm 2: ASPIRE Dataflow

```
1  for ($p_k = 0;\ p_k < K;\ p_k += P_k$)
2   for ($p_e = 0;\ p_e < E;\ p_e += P_e$)
3    for ($p_f = 0;\ p_f < F;\ p_f += P_f$)
4     parallel_for ($k = p_k;\ k < min\,(K,\ p_k + P_k)\,;\ k++$)
5      parallel_for ($e = p_e;\ e < min\,(E,\ p_e + P_e)\,;\ e++$)
6       parallel_for ($f = p_f;\ f < min\,(F,\ p_f + P_f)\,;\ f++$)
7        for ($c = 0;\ c < C;\ c++$)
8         for ($r = 0;\ r < R;\ r++$)
9          for ($s = 0;\ s < S;\ s++$)
10          $O\,[k\,e\,f] += I\,[r+e-1\; s+f-1\; c] \times W\,[k\,r\,s\,c]$
```

We can also derive the overall data footprint $\mathcal{D}$ of each invocation of operations shown in Algorithm 2, which is the summation weight data footprint $C \times R \times S \times P_k$, input data footprint $C \times (R + P_e - 1) \times (S + P_f - 1)$, and psum data footprint $P_e \times P_f \times P_k$. The value $\mathcal{D}$ will be utilized to allocate the shared GLB on-chip memory resource.

IV. ASPIRE ARCHITECTURE

A. Architecture Overview

Figure 4 describes the ASPIRE architecture with $M \times M = 16$ PEs in the whole system and $N \times N = 4$ PEs in each partition. ASPIRE includes a uniform GLB [40] with capacity $\mathcal{G}$ as an intermediate memory hierarchy between off-chip memory and PEs. The auxiliary function unit is responsible for operations such as pooling, activation, and normalization. Since these operations are often fast to process [41], [42], we focus on accelerating convolutional and general matrix multiply (GEMM) operations here. The GLB is equipped with M^2/N^2 transmitter sets, each including $2 \times N$ transmitters, for GLB-to-PE communication. The GLB is also equipped with M^2/N^2 receiver sets, each including N receivers, for PE-to-GLB communication. Each PE is equipped with one transmitter and two receivers for transmitting and receiving data, respectively. The multi-DNN controller (MTC) performs allocation of PE partitions and GLB and correspondingly tunes the MRRs at runtime, based on the per-task progress information and system status.

B. Waveguide & Wavelength Allocation

There are $2 \times M^2/N^2$ waveguides in the ASPIRE architecture. The first set of M^2/N^2 waveguides ($Waveguide$ 0-3 in Figure 4) is used for GLB-to-PE communication or forwarding unmodulated light to corresponding PE partitions for PE-to-GLB communication. The second set of M^2/N^2 waveguides

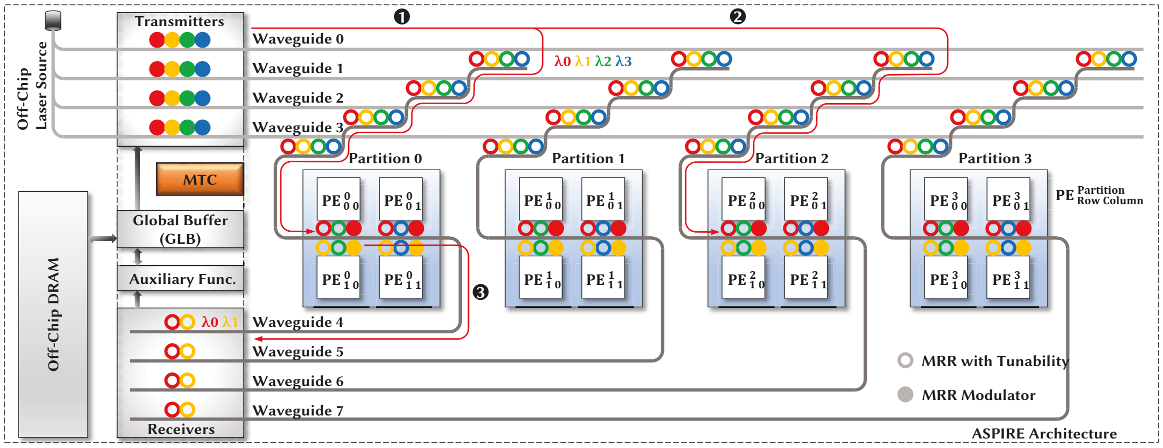

Fig. 4. ASPIRE architecture overview with 4 partitions and 4 PEs per partition. $Waveguide$ 0-3 and 4-7 are implemented for GLB-to-PE and PE-to-GLB communications, respectively. Wavelengths $\lambda0$ - $\lambda3$ are utilized for GLB-to-PE communication while wavelengths $\lambda0$ - $\lambda1$ are reused for PE-to-GLB communication.

($Waveguide$ 4-7 in Figure 4) is solely used for PE-to-GLB communication. Please note that each PE partition can receive data from the GLB via any waveguide in the first set while only sending data to the GLB via a specific waveguide in the second set. For example, $Partition$ 0 can receive data from the GLB via $Waveguide$ 0-3 while only sending data to the GLB via $Waveguide$ 4.

There are $2 \times N$ wavelengths utilized in the ASPIRE architecture. The first set of N wavelengths ($\lambda0$ - $\lambda1$ in Figure 4) is used for row-wise GLB-to-PE multicast communication within each PE partition or across several partitions, as well as PE-to-GLB unicast communication. The second set of N wavelengths ($\lambda2$ - $\lambda3$ in Figure 4) is solely used for column-wise GLB-to-PE multicast communication within each PE partition or across several partitions. Within each PE partition, one receiver attached to each PE in a row resonates on a wavelength from the first set (e.g., $\lambda0$ for PE_{00}^{0} and PE_{01}^{0}) while the other receiver attached to each PE in a column resonates on another wavelength from the second set (e.g., $\lambda2$ for PE_{00}^{0} and PE_{10}^{0}). We can derive that the GLB bandwidth scales with M^2/N. When maintaining the number of PEs in a partition N^2 fixed, the GLB bandwidth increases linearly with the number of PEs in the whole system M^2.

C. Photonic Network

The ASPIRE photonic network can be dynamically configured to support seamless communication between fragmented PE partitions allocated to each inference task. We described how the proposed photonic network is configured in Table I to support the three working cases shown in Figure 5. In Table I, "✓" and "✗" represent the corresponding MRRs are tuned to on-resonant and off-resonant states, respectively. Split ratios of 1/0 and 0/1 are equivalent to on-resonant and off-resonant states, respectively.

CASE I: In the case shown in Figure 5 (a) where each PE partition works independently as a sub-accelerator, a PE partition receives data from the GLB via a designated waveguide and transmits data to the GLB via another designated waveguide. For example, as shown in Figure 4 ❶, $Partition$ 0 receives data via $Waveguide$ 0. As a result, the 4 MRRs corresponding to $Parittion$ 0 and $Waveguide$ 0 and working on wavelengths $\lambda0$ - $\lambda3$ are tuned to the on-resonant state to forward the data. The MRRs corresponding to $Partition$ 0 and other waveguides are tuned to the off-resonant state as these waveguides carry data for other partitions. The MRRs corresponding to other partitions and $Waveguide$ 0 are also tuned to the off-resonant state as these partitions receive data from $Partition$ 0. Within $Partition$ 0, data is multicast to different rows using different wavelengths. For example, PE_{00}^{0} and PE_{01}^{0} receive the same data using wavelength $\lambda0$. Hence, the MRRs working on $\lambda0$ and corresponding to PE_{00}^{0} and PE_{01}^{0} are tuned to split ratios of 1/1 and 1/0, respectively. Similarly, PE_{10}^{0} and PE_{11}^{0} receive the same data using wavelength $\lambda1$. Hence, the MRRs working on $\lambda1$ and corresponding to PE_{10}^{0} and PE_{11}^{0} are tuned to split ratios of 1/1 and 1/0, respectively. Data is also multicast to different columns using different wavelengths. For example, PE_{00}^{0} and PE_{10}^{0} receive the same data using wavelength $\lambda2$. Hence, the MRRs working on $\lambda2$ and corresponding to PE_{00}^{0} and PE_{10}^{0} are tuned to split ratios of 1/1 and 1/0, respectively.

As shown in Figure 4 ❸, $Partition$ 0 transmits data via $Waveguide$ 4. As a result, the 4 MRRs corresponding to $Partition$ 0 and $Waveguide$ 0 and working on wavelengths $\lambda0$ - $\lambda3$ are tuned to the on-resonant state to forward the unmodulated light to $Partition$ 0 for data transmission. The MRRs corresponding to other partitions or waveguides are tuned to the off-resonant state. Within $Partition$ 0, PE_{00}^{0} and PE_{10}^{0} transmit data via $Waveguide$ 4 using wavelengths $\lambda0$ and $\lambda1$, respectively. PE_{01}^{0} and PE_{11}^{0} transmit data via $Waveguide$ 4 using wavelengths $\lambda0$ and $\lambda1$, respectively, in a different time slot. The PE-to-GLB unicast communication is done sequentially in each row of PEs within a partition. Hence, a one-bit token is propagated circularly between columns to determine which column has access to the PE-to-GLB unicast

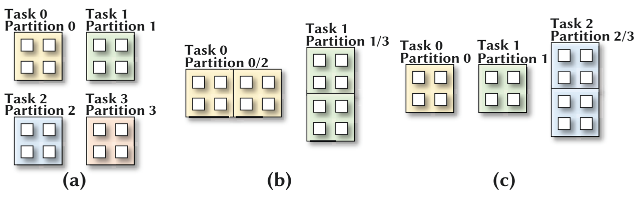

Fig. 5. Aspire working cases: (a) $Partition$ 0-3 are allocated for $Task$ 0-3; (b) $Partition$ 0 & 2 and $Partition$ 1 & 3 are allocated for $Task$ 0 and $Task$ 1, respectively; (c) $Partition$ 0, $Partition$ 1, and $Partition$ 2 & 3 are allocated for $Task$ 0, $Task$ 1, and $Task$ 2, respectively.

TABLE I
MRR Working State in Aspire Architecture

	Partition 0		Partition 1		Partition 2		Partition 3	
	$\lambda0/\lambda1$	$\lambda2/\lambda3$	$\lambda0/\lambda1$	$\lambda2/\lambda3$	$\lambda0/\lambda1$	$\lambda2/\lambda3$	$\lambda0/\lambda1$	$\lambda2/\lambda3$
Case shown in Figure 5 (a)								
Waveguide 0	✓	✓	✗	✗	✗	✗	✗	✗
Waveguide 1	✗	✗	✓	✓	✗	✗	✗	✗
Waveguide 2	✗	✗	✗	✗	✓	✓	✗	✗
Waveguide 3	✗	✗	✗	✗	✗	✗	✓	✓
Case shown in Figure 5 (b)								
Waveguide 0	1/1	✓	✗	✗	✓	✗	✗	✗
Waveguide 1	✗	✗	✓	1/1	✗	✗	✗	✓
Waveguide 2	✗	✗	✗	✗	✗	✓	✗	✗
Waveguide 3	✗	✗	✗	✗	✗	✗	✓	✗
Case shown in Figure 5 (c)								
Waveguide 0	✓	✓	✗	✗	✗	✗	✗	✗
Waveguide 1	✗	✗	✓	✓	✗	✗	✗	✗
Waveguide 2	✗	✗	✗	✗	✓	1/1	✗	✓
Waveguide 3	✗	✗	✗	✗	✗	✗	✓	✗

communication channel. We do not show the token propagation network in Figure 4 for simplicity.

Please note that, in this case, all partitions can work independently without any interference because communication between each partition and the GLB is done by separate waveguides. It is also unnecessary to synchronize the operations on different partitions. For example, it is possible that $Partition$ 0 is loading data from the GLB through $Waveguide$ 0 while $Partition$ 1 is sending data to the GLB using $Waveguide$ 1 and 5.

CASE II/III: In the case shown in Figure 5 (b) and (c) where several PE partitions work jointly as a sub-accelerator, these partitions are likely to receive data from the GLB via a designated waveguide and transmit data to the GLB via separate waveguides. For example, as shown in Figure 5 (b), $Partition$ 0 and $Partition$ 2 are horizontally combined as a sub-accelerator. They receive row-wise multicast data from $Waveguide$ 0 but receive column-wise multicast data from $Waveguide$ 0 and $Waveguide$ 2, respectively. To receive row-wise multicast data from $Waveguide$ 0, the 2 MRRs corresponding to $Partition$ 0 and $Waveguide$ 0 and working on wavelengths $\lambda0$ - $\lambda1$ are tuned to the split ratio of 1/1 to forward half of the light power to $Partition$ 0. Meanwhile, the 2 MRRs corresponding to $Partition$ 2 and $Waveguide$ 0 and working on wavelengths $\lambda0$ - $\lambda1$ are tuned to the on-resonant state to forward the rest of light power to $Partition$ 2, as shown in Figure 4 ❷. The states of other MRRs outside each partition can be similarly derived. We summarize the state of each MRR in all three working cases in Table I.

V. Aspire Allocation Algorithm

Allocation of the PE partitions and GLB capacity is invoked in either the presence of a new inference task or the completion of an existing inference task. The MTC shown in Figure 4 keeps track of the concurrent inference tasks. An entry is created upon arrival of a new inference task using `CreateEntry()` ($Line$ 17-20 in Algorithm 3). The arrival time is recorded as well as the overall number of MAC operations and the isolate execution time, which can be obtained offline. An entry is deleted upon completion of an existing inference task using `DeleteEntry()` ($Line$ 8-15 in Algorithm 3). The MTC constantly monitors and updates the progress of the concurrent inference tasks using `UpdateEntry()` ($Line$ 6 in Algorithm 3). An allocation is invoked after `request` signal is set while the implementation of the previous allocation is complete (`allocation` signal is cleared). The necessary operations (e.g., MRR tuning) to implement the current allocation are pushed in the `Ctrl` queue to be performed asynchronously. Inference tasks losing one or several partitions in the allocation can release the partition(s) at the end of the current invocation and continue the next updated one without interruptions. By contrast, inference tasks gaining one or more partitions in the allocation have to wait until all added partition(s) are released. In this way, we minimize the timing overhead of updating the allocation of the PE partitions and GLB capacity.

`AllocationAlgorithm()` is the key function of the proposed allocation algorithm. It includes the allocation of PE partitions $\mathcal{S}_i$, allocation of parallelism in three dimensions $\langle P_k^i\, P_e^i\, P_f^i \rangle$, and allocation of GLB capacity $\mathcal{G}_i$ for $Task\ i$. The MTC keeps track of the following parameters and feeds them to the allocation algorithm:

$T_{isolate}^i$: The execution time of $Task\ i$ when it is processed on Aspire alone without interruptions. This value can be obtained offline.
T_{remain}^i: An implication of computing resource demand which is defined by Equation (1).
$T_{deadline}^i$: The difference between the expected completion time and current time. It is within the range of $\left(-\infty,\, SLA \times T_{isolate}^i\right]$.

$$T_{remain}^i = \frac{No.\,of\,remained\,MACs}{No.\,of\,total\,MACs} \times T_{isolate}^i \qquad (1)$$

The allocation of PE partitions $\mathcal{S}_i$ is derived from Equation (2) which holistically considers the computing resource demand represented by T_{remain}^i and task priority represented by $e^{-T_{deadline}^i}$. Compared to the linear expression of task priority in prior work [7], [8], the proposed exponential expression can assign proper priorities to tasks that have passed their deadlines ($T_{deadline}^i < 0$) and significantly favor tasks with tight deadlines, facilitating equal progress of tasks and adherence to SLA.

Algorithm 3: ASPIRE Allocation Algorithm

```
function ALLOCATIONSCHEME (count, LIST, TASK)
    request = false
    allocation = false
    for cycle ← [0 : ∞) do
        // update information of existing DNN inference tasks
        UPDATEENTRY (count, LIST)
        // check completion of any existing DNN inference task
        if count ≠ -1 then
            for i ← [0 : count] do
                if LIST[i] . remain = 0 then
                    DELETEENTRY (i, count, LIST)
                    request = true
                end if
            end for
        end if
        // looking for newly arrived tasks
        while TASK ≠ NULL do
            CREATEENTRY (Deq(TASK), count, LIST)
            request = true
        end while
        // initiate allocation algorithm
        if request = true then
            if allocation = false then
                Enq(CTRL, ALLOCATIONALGORITHM (count, LIST))
                allocation = true
                request = false
            end if
        end if
        // test completion of allocation
        if CTRL = NULL then
            allocation = false
        end if
        cycle ++
    end for
end function
```

$$\mathcal{S}_i = \frac{M^2}{N^2} \times \frac{T^i_{remain} \times e^{-T^i_{deadline}}}{\sum T^i_{remain} \times e^{-T^i_{deadline}}} \quad (2)$$

The parallelism in three dimensions $\langle P^i_k\, P^i_e\, P^i_f \rangle$ is determined through heuristic search with the target of minimizing GLB accesses and the constraint of $\lceil P_k/N \rceil \times \lceil P_e \times P_f/N \rceil \leq \mathcal{S}_i$. The allocation of GLB capacity $\mathcal{G}_i$ is derived from equation $\mathcal{G}_i = \mathcal{D}_i \times \mathcal{G}/\sum \mathcal{D}_i$, where $\mathcal{D}_i$ is defined in Section 3 and $\mathcal{G}$ is the overall GLB capacity. In the case of $\mathcal{G}_i < \mathcal{D}_i$, tiling along $\langle c \rangle$ dimension is performed.

VI. Evaluation Methodology

We extend the open-source SCALE-Sim simulator [51] to support temporal and spatial multiplexing of heterogeneous DNN inference tasks. The execution time includes time for both computation and communication. The extended simulator can track the number of MAC operations and the number of accesses to each memory hierarchy (local register, GLB, and off-chip memory), from which the time for both computation and communication is derived. The derivation process has taken the interference between multiple existing DNN inference tasks into consideration and enforced the bandwidth limitation of each metallic-based or photonic interconnects. The delay for tuning MRRs is set to 500 ps [52], indicating that DNN inference tasks with sufficient partitions can progress without being interrupted by this tuning process. Similarly, the waiting tasks can immediately start progress upon obtaining sufficient partitions. Table II lists the key architecture parameters of ASPIRE for simulation. We utilize similar parameters as in [5], [7] for a fair comparison. We assume 1×10^{-12} bit error rate (BER) [49] at 10 $Gbps$ bit rate and wavelengths around 1550 nm. We also assume a maximum free spectral range (FSR) limit of 50 nm [22] which reflects the fabrication limitation of MRR radius. As the maximum number of wavelengths assumed in ASPIRE is 64, the power penalty due to crosstalk is negligible due to prior study [53].

TABLE II
ASPIRE SIMULATION PARAMETERS

Parameter	Value
PE Array Dimension	128×128
PE Partition Dimension	8×8 8×16 16×16 16×32 32×32
PE Operating Frequency	700 MHz
Global Buffer (GLB)	12 MB
Data Width	8-bit Weight & Input Feature 16-bit Psum
GLB Read Bandwidth	1280 GB/s
GLB Write Bandwidth	640 GB/s
GLB Access Latency	1 cycle
Off-chip Memory Model	DDR3 (34 GB/s) HBM (128/256/358/512 GB/s)

TABLE III
SIMULATION PHOTONIC PARAMETERS

Component	Value	Component	Value
Laser Source	5 dB [43]	Ring Drop	0.7 dB [44]
Coupler	1 dB [43]	Ring Through	0.01 dB [45]
Splitter	0.2 dB [46]	Photodetector	0.1 dB [43]
Waveguide	1 dB/cm [43]	Waveguide-to-receiver	0.5 dB [47]
Waveguide Bend	0.01 dB [48]	Receiver Sensitivity	-23.4 dBm [49]
Waveguide Crossover	0.05 dB [47]	Ring Heating	320 μW [50]

A. Power Model

The power consumption value of MAC operations is obtained using Synopsys Design Compiler. The power consumption values of access local buffers and the GLB are obtained using CACTI 6.0 [54] while the power consumption value of accessing off-chip memory is obtained using DRAMSim2 [55]. The power consumption value of metallic-based interconnects is obtained using DSENT [44] and parameters in [56]. The power consumption value of photonic interconnects is derived from Equation (3) shown below:

$$P_{total} = P_{TX} + P_{RX} + P_{laser} + P_{thermal} \quad (3)$$

The total power consumption P_{total} consists of 4 parts: the power consumption of transmitters P_{TX}, the power consumption of receivers P_{RX}, the power consumption of the off-chip laser source P_{laser}, and the power consumption of resistive heaters for mitigating thermal and process variations of MRRs $P_{thermal}$. P_{TX} and P_{RX} are derived from parameters in [50]. The power consumption values for each transmitter and receiver are 0.9 mW and 0.6 mW, respectively. The power consumption for thermal heating is assumed to be 0.32 mW per MRR [50]. The power consumption of the off-chip laser source is derived from Equation (4) shown below:

$$P_{laser} = P_{rs} + C_{loss} + P_{extinction} + M_{system} \tag{4}$$

The power consumption of the off-chip laser source consists of 4 parts: the photodetector sensitivity P_{rs}, the overall insertion loss C_{loss} which can be derived from parameters listed in Table III, the power penalty caused by extinction ratio $P_{extinction}$ (2 dB [57]), and the system margin M_{system} (4 dB [58]). The purpose of the system margin is to allocate an amount of power to additional power penalty sources developed during the system's lifetime.

B. *Multi-DNN Accelerators*

The proposed ASPIRE (AS) architecture is compared against two temporal multi-DNN accelerators named PREMA [5] (PR) and AI-MT [6] (AI), and two spatial multi-DNN accelerators named Planaria [7] (PL) and Dataflow Mirroring (DM) [8]. For a fair comparison, these multi-DNN accelerators and ASPIRE are all scaled to include $M \times M = 16384$ PEs and $N \times N = 1024$ PEs per partition unless otherwise stated. The GLB size is set to 12 MB [5], [7]. Weights and inputs are assumed to be 8-bit wide while psums are assumed to be 24-bit wide [9]. The PE clock frequency is set to 700 MHz. We perform cycle-level memory simulation [55] by assuming a high bandwidth memory (HBM) module with 358 GB/s bandwidth as in [5], [7]. We also explore the impact of memory bandwidth on system performance by implementing a DDR3 memory model as in [1] and varying HBM bandwidth as shown in Table II.

C. *Multi-DNN Benchmark*

We generate a multi-DNN benchmark from eleven convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformers in four categories: (1) EfficientNet-b0 [36], EfficientNet-b7 [36], MobileNet-v1 [37], DenseNet-201 [33], ResNet=50 [34], and VGG-16 [32] using ImageNet dataset for image classification; (2) YOLO-v3 [31] and SSD-MobileNet-v1 [37] using COCO dataset for object detection; (3) GNMT [37] using WMT E-G dataset for translation; (4) BERT [37] using SQuAD v1.1 dataset for natural language processing. A certain sequence of the mixed DNN models is generated and applied to simulation on both ASPIRE architecture and other multi-DNN accelerators. Please note that the sequence is long enough to model diverse task multiplexing cases. The arrival time of each DNN inference task in the sequence is randomly assigned from a Poisson distribution as in [7]. The task arrival rate can be adjusted by tuning the rate parameter λ. We evaluate ASPIRE architecture performance under different task arrival rates in our simulation. Please note that only the convolutional and fully-connected layers are considered in our simulation.

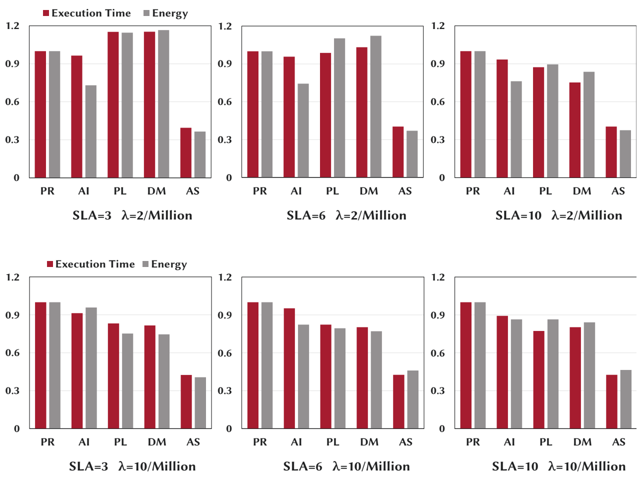

Fig. 6. Execution time and energy consumption comparison between ASPIRE and other multi-DNN accelerators when varying SLA and task arrival rate λ.

D. *Evaluation Metrics*

We utilize the following four metrics to evaluate ASPIRE architecture and other multi-DNN accelerators. The first two metrics are from the perspective of cloud vendors while the last two metrics are from the perspective of end users.

- *Execution Time*: the time spent to process the above sequence of mixed DNN inference tasks.
- *Energy Consumption*: the energy consumption of processing the above sequence of mixed DNN inference tasks.
- *SLA Satisfaction Rate*: the fraction of multiplexed DNN inference tasks that adhere to the SLA profile. SLA of $Task$ i is defined as $T^i_{deadline}/T^i_{isolate}$.
- *Fairness*: the measurement of the equal progress of the multiplexed DNN inference tasks [59].

VII. EXPERIMENT RESULTS

A. *Execution Time & Energy Consumption*

We model three different SLA values SLA=3, SLA=6, and SLA=10, which represent cases with strict, medium, and relaxed deadlines, respectively. We also model two different task arrival rate λ values $\lambda = 2/Million$ and $\lambda = 10/Million$, which represent cases with low and high task arrival rates. $\lambda = 2/Million$ means on average 2 DNN inference tasks are expected in a one-million clock cycle interval. Figure 6 shows the execution time and energy consumption comparison between ASPIRE and other multi-DNN accelerators. All values are normalized to PR. We make the following observations. First, in the cases with strict or medium deadlines (SLA=3 or SLA=6) and low task arrival rate $\lambda = 2/Million$, spatial multi-DNN accelerators PL and DM perform worse than temporal multi-DNN accelerators PR and AI because the chance of spatially co-executing several inference tasks is low in such cases. As a result, the spatial task parallelism capability in PL and DM is not fully leveraged while the control overhead still exists. Second, in other cases with a high chance of spatially co-executing several inference tasks, PL and DM perform better

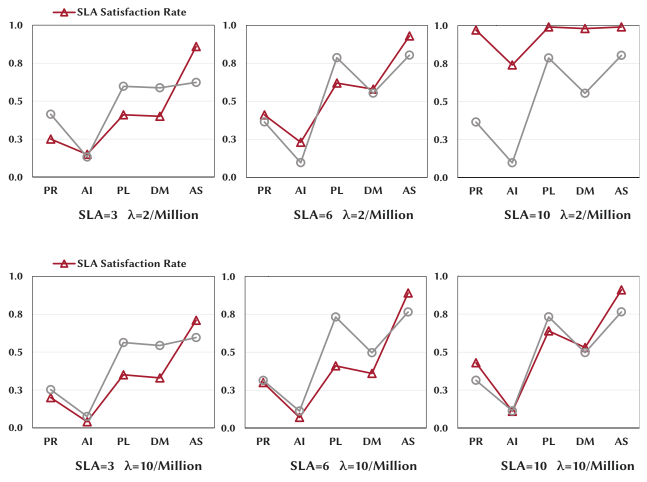

Fig. 7. SLA satisfaction rate and fairness comparison between ASPIRE and other multi-DNN accelerators when varying SLA and task arrival rate λ.

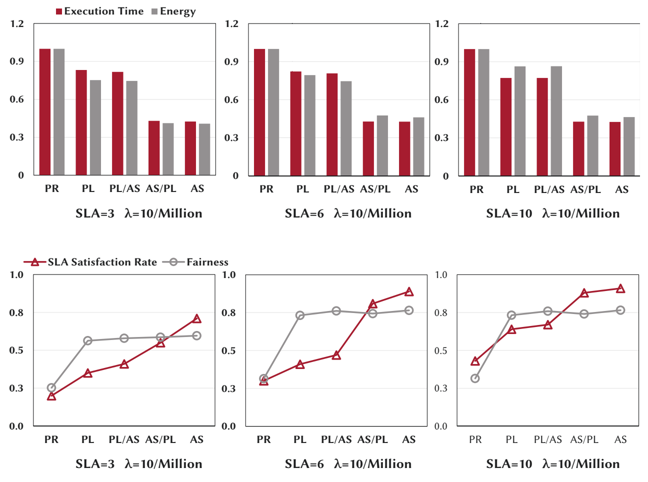

Fig. 8. ASPIRE architecture and allocation algorithm benefits breakdown analysis. All values are normalized to PR.

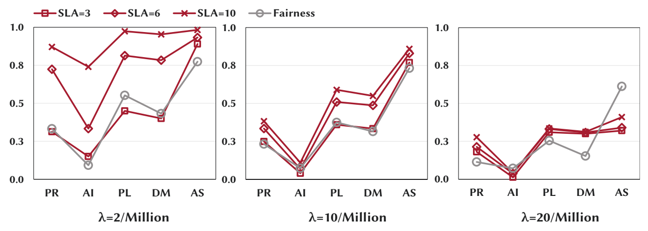

Fig. 9. ASPIRE allocation algorithm compared with other multi-DNN accelerators when randomly assigning SLA values to incoming inference tasks.

than PR and AI in terms of both execution time and energy consumption. Third, ASPIRE outperforms other multi-DNN accelerators mainly due to the proposed photonic network design. In particular, ASPIRE achieves up to 61% and 69% reduction in execution time and energy consumption.

B. SLA Satisfaction Rate & Fairness

Figure 7 shows the SLA satisfaction rate and fairness comparison between ASPIRE and other multi-DNN accelerators. All values are normalized to PR. We make the following observations. First, in cases with different combinations of SLA values and task arrival rates λ, temporal multi-DNN accelerators (PR and AI) achieve lower SLA satisfaction rates and fairness compared to spatial multi-DNN accelerators (PL, DM, and ASPIRE), as temporal accelerators often have relatively low PE utilization and cannot delicately allocate hardware resource to multiple inference tasks. In particular, AI achieves as low as 4% SLA satisfaction rate and 0.08 fairness because it prompts balancing of communication- and computation-intensive tasks over the QoS concern. Second, the SLA satisfaction rate of each accelerator tends to decrease when adopting a more strict task deadline or high task arrival rate. Third, ASPIRE still maintains a relatively high SLA satisfaction rate of 71% and fairness of 0.60 in the case with strict task deadline (SLA=3) and high task arrival rate ($\lambda = 10/Million$). This is due to the combined effects of the photonic network design and partition allocation strategy.

C. ASPIRE Contribution Breakdown Analysis

We present the contributions of the ASPIRE photonic network design and allocation strategy to the overall system improvement separately in Figure 8. We only show the case with critical SLA (SLA=3) and high task arrival rate ($\lambda = 10/Million$). ASPIRE is compared against PR, PL, and two additional baselines PL/AS and AS/PL. PL/AS is a design that combines PL architecture and ASPIRE allocation algorithm, while AS/PL is a design that combines ASPIRE architecture and PL allocation algorithm. We make the following two observations. First, as compared to PL, PL/AS achieves a minor reduction in execution time (1.5%) and energy consumption (1.0%) while AS/PL achieves a significant reduction in execution time (48.3%) and energy consumption (43.2%), indicating that the reduction in execution time and energy consumption of ASPIRE mainly comes from the proposed photonic network design. Second, both PL/AS and AS/PL achieve significant improvement in SLA satisfaction rate and fairness, as compared to PL, indicating that both the proposed photonic network architecture and the allocation strategy help the co-executed tasks to achieve equal progress and fulfill the deadlines.

D. Dynamic Task Deadline

In ASPIRE, the initial value of $T_{deadline}$ is considered an implicit priority parameter and will be continuously updated. To demonstrate the benefits of the implicit priority parameter in ASPIRE over the explicit priority parameter in PL and DM, we randomly assign one of the three SLA values (SLA=3, SLA=6, and SLA=10) to each inference task. Meanwhile, since PL and DM adopt an explicit priority in the range of 1 to 11 [7], we assign one of the three priority values (8, 5, and 1) to each corresponding inference task. Figure 9 shows the SLA satisfaction rate of different groups of tasks, each group corresponds to a specific priority (e.g., implicit priority SLA=3 in ASPIRE and explicit priority 8 in PL and DM). We observe

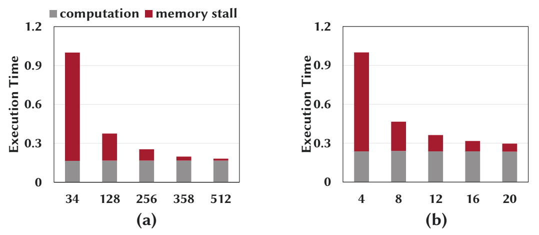

Fig. 10. GLB (a) bandwidth and (b) capacity exploration assuming SLA=3 and $\lambda = 10/Million$.

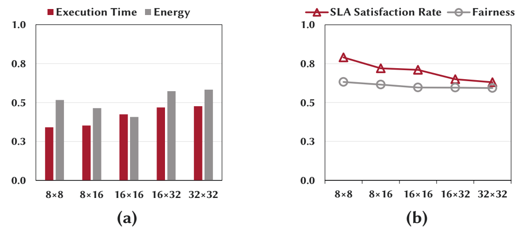

Fig. 11. Partition size exploration assuming SLA=3 and $\lambda = 10/Million$.

that in general inference tasks with high priority, either implicit or explicit, achieve higher SLA satisfaction rates under different task arrival rate assumptions. Furthermore, the implicit priority in ASPIRE is more effective in terms of SLA satisfaction rate and fairness, as compared to the explicit priority in PL and DM due to the fact that ASPIRE allocation strategy significantly prompts tasks that are close to or have passed their deadlines. We show the results of an extreme case with a very high task arrival rate ($\lambda = 20/Million$) in which many tasks cannot be completed in time. ASPIRE still achieves high fairness in this case compared to other designs.

E. GLB Bandwidth and Capacity Exploration

Figure 10 (a) shows the impact of bandwidth between main memory and GLB on execution time in ASPIRE, when assuming SLA=3 and $\lambda = 10/Million$. We explore several setups, DDR3 off-chip memory with 34 GB/s bandwidth as in [1], HBM with 128-512 GB/s bandwidth (PR [5] and PL [7] assume HBM with 358 GB/s bandwidth). The execution time is further divided into the time for computation and memory stall time. We observe that 358 GB/s bandwidth is sufficient as the memory stall time only accounts for 1.8% of the overall execution time. As the bandwidth value decreases, the fraction of memory stall time in overall execution time increases (83% in DDR3 with 34 GB/s bandwidth). Meanwhile, further increasing the memory bandwidth over 358 GB/s only leads to an insignificant reduction in memory stall time. We also explore the impact of GLB capacity on execution time in ASPIRE as shown in Figure 10 (b). Implementing a 12 MB GLB would incur 31% memory stall time, which can be reduced to 21% when implementing a 20 MB GLB.

F. Partition Size Exploration

We study the execution time and energy consumption, as well as SLA satisfaction rate and fairness when varying the partition size in ASPIRE architecture. The study is performed assuming the critical SLA and high task arrival rate. As shown in Figure 11, we observe increasing execution time as partition size increases, which indicates that a smaller partition size can better facilitate the process of allocating the right amount of hardware resource to a particular inference task. By contrast, the minimum energy consumption is achieved when assuming 16×16 partition size. Though partition size does not affect the energy consumption of computation and data access to different memory hierarchies, it has a profound impact on the energy consumption of the proposed photonic network. Small partition sizes (8×8 and 8×16) lead to a large number of MRRs required in the photonic network and high energy consumption for thermal tuning. Large partition sizes (16×32 and 32×32) lead to high insertion loss and high energy consumption for the off-chip laser source. We observe that the proposed ASPIRE architecture achieves optimal energy-delay product value when assuming 16×16 partition size. We can observe that the SLA satisfaction rate and fairness decrease as the partition size increases, which also indicates that a smaller partition size can fully exploit the spatial task parallelism on the accelerator and improve QoS and fairness.

G. Area Estimation

We synthesize ASPIRE design using `Synopsys Design Compiler` and 28 nm technology. The size of the 12 MB GLB is 89.55 mm^2 while the area overhead of the photonic network (transceivers and MRRs) is 46.06 mm^2. Since ASPIRE is not equipped with the 1 MB weight FIFO (3.92 mm^2 and 2 MB Accumulation Buffer (20.43 mm^2), we can mitigate the area overhead of the photonic network by using an 8 MB GLB in ASPIRE. According to Figure 6 and Figure 10 (b), such a system setup can still achieve lower execution time as compared to PL with 12 MB GLB.

VIII. CONCLUSION

In this paper, we propose a photonic multi-DNN accelerator named ASPIRE that supports spatial and temporal co-execution of multiple DNNs. The salient features of ASPIRE architecture include (1) a novel dataflow that converts a sufficient fraction of data reuse opportunities into data multicast opportunities to leverage the energy-efficient multicast property of silicon photonics; (2) a photonic network that can be adaptively divided into several sub-networks, each seamlessly connecting some fragmented PE partitions to construct a sub-accelerator for one DNN; (3) an algorithm that produces the allocation of PE partitions based on computing demand and task priority, and the allocation of on-chip memory capacity based on DNN topology and data reuse pattern. Simulation studies using a collection of diverse CNN, RNN, and transformer models have proven the effectiveness of ASPIRE architecture compared to other state-of-the-art multi-DNN accelerators.

IX. ACKNOWLEDGEMENTS

This research was partially supported by NSF grants CCF-1901165, CCF-2311543, CCF-1953980, CCF-1703013, CCF-1901192, CCF-1936794, and CCF-2311544. We sincerely thank the anonymous reviewers for their excellent feedback.

References

[1] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. luc Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 1–12.

[2] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A Configurable Cloud-Scale DNN Processor for Real-Time AI," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2018, pp. 1–14.

[3] U. Gupta, S. Hsia, V. Saraph, X. Wang, B. Reagen, G.-Y. Wei, H.-H. S. Lee, D. Brooks, and C.-J. Wu, "DeepRecSys: A System for Optimizing End-to-End At-Scale Neural Recommendation Inference," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, May 2020, pp. 982–995.

[4] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "INFaaS: A Model-Less and Managed Inference Serving System," *arXiv Preprint*, pp. 1–16, May 2019.

[5] Y. Choi and M. Rhu, "PREMA: A Predictive Multi-Task Scheduling Algorithm for Preemptible Neural Processing Units," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, February 2020, pp. 220–233.

[6] E. Baek, D. Kwon, and J. Kim, "A Multi-Neural Network Acceleration Architecture," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, May 2020, pp. 940–953.

[7] S. Ghodrati, B. H. Ahn, J. K. Kim, S. Kinzer, B. R. Yatham, N. Alla, H. Sharma, M. Alian, E. Ebrahimi, N. S. Kim, C. Young, and H. Esmaeilzadeh, "Planaria: Dynamic Architecture Fission for Spatial Multi-Tenant Acceleration of Deep Neural Networks," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, October 2020, pp. 681–697.

[8] J. Lee, J. Choi, J. Kim, J. Lee, and Y. Kim, "Dataflow Mirroring: Architectural Support for Highly Efficient Fine-Grained Spatial Multitasking on Systolic-Array NPUs," in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, December 2021, pp. 247–252.

[9] Y. S. Shao, J. Clemons, R. Venkatesan, B. Zimmer, M. Fojtik, N. Jiang, B. Keller, A. Klinefelter, N. Pinckney, P. Raina, S. G. Tell, Y. Zhang, W. J. Dally, J. Emer, and C. T. Gray, "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-based Architecture," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, October 2019, pp. 14–27.

[10] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 367–379.

[11] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting Vision Processing Closer to the Sensor," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2015, pp. 92–104.

[12] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A Dynamically Configurable Coprocessor for Convolutional Neural Networks," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2010, pp. 247–257.

[13] L. Cavigelli, D. Gschwend, C. Mayer, S. Willi, B. Muheim, and L. Benini, "Origami: A Convolutional Network Accelerator," in *Proceedings of the ACM Great Lakes Symposium on VLSI (GLVLSI)*, May 2015, pp. 199–204.

[14] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning," in *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, February 2014, pp. 269–284.

[15] J. Albericio, P. Judd, T. Hetherington, T. Adamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 1–13.

[16] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An Accelerator for Compressed-Sparse Convolutional Neural Networks," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2017, pp. 27–40.

[17] Y. Li, A. Louri, and A. Karanth, "SPACX: Silicon Photonics-based Scalable Chiplet Accelerator for DNN Inference," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, April 2022, pp. 831–845.

[18] Y. Li, K. Wang, H. Zheng, A. Louri, and A. Karanth, "ASCEND: A Scalable and Energy-Efficient Deep Neural Network Accelerator with Photonic Interconnects," *IEEE Transactions on Circuits and Systems I (TCAS-I)*, vol. 69, no. 7, pp. 2730–2741, July 2022.

[19] Y. Li, A. Louri, and A. Karanth, "Scaling Deep-Learning Inference with Chiplet-based Architecture and Photonic Interconnects," in *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, December 2021, pp. 931–936.

[20] D. A. B. Miller, "Device Requirements for Optical Interconnects to Silicon Chips," *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1166–1185, June 2009.

[21] R. Soref, "The Past, Present, and Future of Silicon Photonics," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 12, no. 6, pp. 1678–1687, November 2006.

[22] K. Bergman, L. P. Carloni, A. Biberman, J. Chan, and G. Hendry, *Photonic Network-on-Chip Design*. Springer, 2014.

[23] Y. Demir, Y. Pan, S. Song, N. Hardavellas, J. Kim, and G. Memik, "Galaxy: A High-Performance Energy-Efficient Multi-Chip Architecture Using Photonic Interconnects," in *Proceedings of the ACM International Conference on Supercomputing (ICS)*, June 2014, pp. 303–312.

[24] N. Kirman, M. Kirman, R. K. Dokania, J. F. Martinez, A. B. Apsel, M. A. Watkins, and D. H. Albonesi, "Leveraging Optical Technology in Future Bus-based Chip Multiprocessors," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, December 2006, pp. 492–503.

[25] P. Grani, R. Proietti, V. Akella, and S. J. B. Yoo, "Design and Evaluation of AWGR-based Photonic NoC Architectures for 2.5D Integrated High Performance Computing Systems," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, February 2017, pp. 289–300.

[26] D. Vantrease, R. Schreiber, M. Monchiero, M. McLaren, N. P. Jouppi, M. Fiorentino, A. Davis, N. Binkert, R. G. Beausoleil, and J. H. Ahn, "Corona: System Implications of Emerging Nanophotonic Technology," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2008, pp. 153–164.

[27] A. K. Ziabar, J. L. Abellan, R. Ubal, C. Chen, A. Joshi, and D. Kaeli, "Leveraging Silicon-Photonic NoC for Designing Scalable GPUs," in *Proceedings of the ACM International Conference on Supercomputing (ICS)*, June 2015, pp. 273–282.

[28] Y. Pan, P. Kumar, J. Kim, G. Memik, Y. Zhang, and A. Choudhary, "Firefly: Illuminating Future Network-on-Chip with Nanophotonics," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, June 2009, pp. 429–440.

[29] Y. Thonnart, S. Bernabe, J. Charbonnier, C. Bernard, D. Coriat, C. Fuguet, P. Tissier, B. Charbonnier, S. Malhouitre, D. Saint-Patrice, M. Assous, A. Narayan, A. Coskun, D. Dutoit, and P. Vivet, "POPSTAR: A Robust Modular Optical NoC Architecture for Chiplet-based 3D Integrated Systems," in *Porceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2020, pp. 1456–1461.

[30] Y. Li, A. Louri, and A. Karanth, "SPRINT: A High-Performance, Energy-Efficient, and Scalable Chiplet-based Accelerator with Photonic Interconnects for CNN Inference," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 33, no. 10, pp. 2332–2345, October 2022.

[31] J. Redmon and A. Farhadi, "Yolov3: An Incremental Improvement," *arXiv Preprint*, 2018.

[32] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv preprint*, 2014.

[33] G. Huang, Z. Liu, L. V. D. Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[34] 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 (CVPR)*, 2016.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[36] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

[37] V. J. Reddi, C. Cheng, D. Kanter, P. Mattson, G. Schmuelling, C.-J. Wu, B. Anderson, M. Breughe, M. Charlebois, W. Chou, R. Chukka, C. Coleman, S. Davis, P. Deng, G. Diamos, J. Duke, D. Fick, J. S. Gardner, I. Hubara, S. Idgunji, T. B. Jablin, J. Jiao, T. S. John, P. Kanwar, D. Lee, J. Liao, A. Lokhmotov, F. Massa, P. Meng, P. Micikevicius, C. Osborne, G. Pekhimenko, A. T. R. Rajan, D. Sequeira, A. Sirasao, F. Sun, H. Tang, M. Thomson, F. Wei, E. Wu, L. Xu, K. Yamada, B. Yu, G. Yuan, A. Zhong, P. Zhang, and Y. Zhou, "MLPerf Inference Benchmark," in *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, May 2020, pp. 446–459.

[38] R. Marchetti, C. Lacava, L. Carroll, K. Gradkowski, and P. Minzioni, "Coupling Strategies for Silicon Photonics Integrated Chips," *Photonics Research*, vol. 7, no. 2, pp. 201–239, January 2019.

[39] W. Bogaerts, P. D. Heyn, T. V. Vaerenbergh, K. D. Vos, S. K. Selvaraja, T. Claes, P. Dumon, P. Bienstman, D. V. Thourhout, and R. Baets, "Silicon Microring Resonators," *Laser & Photonics Reviews*, vol. 6, no. 1, pp. 47–73, January 2012.

[40] P. Fotouhi, S. Werner, J. Lowe-Power, and S. J. B. Yoo, "Enabling Scalable Chiplet-based Uniform Memory Architectures with Silicon Photonics," in *Proceedings of the International Symposium on Memory Systems (MEMSYS)*, September 2019, pp. 222–234.

[41] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS: Scalable and Efficient Neural Network Acceleration with 3D Memory," in *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2017.

[42] X. Yang, M. Gao, Q. Liu, J. Setter, J. Pu, A. Nayak, S. Bell, K. Cao, H. Ha, P. Raina, C. Kozyrakis, and M. Horowitz, "Interstellar: Using Halide's Scheduling Language to Analyze DNN Accelerators," in *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2020.

[43] R. Morris, A. Karanth, and A. Louri, "Dynamic Reconfiguration of 3D Photonic Networks-on-Chip for Maximizing Performance and Improving Fault Tolerance," in *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2012.

[44] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, May 2012, pp. 201–210.

[45] S. Pasricha and S. Bahirat, "OPAL: A Multi-Layer Hybrid Photonic NoC for 3D ICs," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2011.

[46] S. Werner, J. Navaridas, and M. Lujan, "Designing Low-Power, Low-Latency Networks-on-Chip by Optimally Combining Electrical and Optical Links," in *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, February 2017, pp. 265–276.

[47] R. W. Morris and A. Karanth, "Power-Efficient and High-Performance Multi-Level Hybrid Nanophotonic Interconnect for Multicores," in *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, 2010.

[48] A. Biberman, K. Preston, G. Hendry, N. Sherwood-Droz, J. Chan, J. S. Levy, M. Lipson, and K. Bergman, "Photonic Network-on-Chip Architectures Using Multilayer Deposited Silicon Materials for High-Performance Chip Multiprocessors," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 2011.

[49] H. T. Chen, J. Verbist, P. Verheyen, P. D. Heyn, G. Lepage, J. D. Coster, P. Absil, X. Yin, J. Bauwelinck, J. V. Campenhout, and G. Roelkens, "High Sensitivity 10Gb/s Si Photonic Receiver based on a Low-Voltage Waveguide-Coupled Ge Avalanche Photodetector," *Optics Express*, vol. 23, no. 2, 2015.

[50] A. Joshi, C. Batten, Y.-J. Kwon, S. Beamer, I. Shamim, K. Asanovic, and V. Stojanovic, "Silicon-Photonic Clos Networks for Global On-Chip Communication," in *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, May 2009, pp. 124–133.

[51] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A Systematic Methodology for Characterizing Scalability of DNN Accelerators using SCALE-Sim," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2020, pp. 58–68.

[52] E. Peter, A. Thomas, A. Dhawan, and S. R. Sarangi, "Active Microring Based Tunable Optical Power Splitters," *Optics Communications*, vol. 359, pp. 311–315, January 2016.

[53] K. Padmaraju, N. Ophir, A. Biberman, L. Chen, E. Swan, J. Chan, M. Lipson, and K. Bergman, "Intermodulation crosstalk from silicon microring modulators in wavelength-parallel photonic networks-on-chip," in *Annual Meeting of the IEEE Photonics Society*, 2010.

[54] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A Tool to Model Large Caches," *HP Laboratories*, pp. 1–24, April 2009.

[55] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A Cycle Accurate Memory System Simulator," *IEEE Computer Architecture Letters (CAL)*, vol. 10, no. 1, pp. 16–19, January 2011.

[56] J. M. Wilson, W. J. Turner, J. W. Poulton, B. Zimmer, X. Chen, S. S. Kudva, S. Song, S. G. Tell, N. Nedovic, W. Zhao, S. R. Sudhakaran, C. T. Gray, and W. J. Dally, "A 1.17 pJ/b 25Gb/s/pin Ground-Referenced Single-Ended Serial Link for Off- and On-Package Communication in 16nm CMOS Using a Process- and Temperature-Adaptive Voltage Regulator," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC)*, February 2018, pp. 276–278.

[57] C. DeCusatis, *Handbook of Fiber Optic Data Communication: A Practical Guide to Optical Networking*. Academic Press, 2013.

[58] A. V. Krishnamoorthy, R. Ho, X. Zheng, H. Schwetman, J. Lexau, P. Koka, G. Li, I. Shubin, and J. E. Cunningham, "Computer Systems Based on Silicon Photonic Interconnects," *Proceedings of the IEEE*, vol. 97, no. 7, pp. 1337–1361, July 2009.

[59] S. Eyerman and L. Eeckhout, "System-Level Performance Metrics for Multiprogram Workloads," *IEEE Micro*, 2008.