HPPI: A High-Performance Photonic Interconnect Design for Chiplet-Based DNN Accelerators

Guanglong Li and Yaoyao Ye, *Member, IEEE*

***Abstract*—In pursuit of higher inference accuracy, the complexity and parameter size of recent deep neural networks (DNNs) have increased significantly. Due to the increasing demand for computing power, the chiplet-based accelerator has been an important computing platform that can handle these DNN models more efficiently. In widely used DNN models, the feature sizes and the number of channels vary greatly among different convolutional layers. Existing chiplet-based accelerators typically adopt consistent optimization strategy for all of the convolutional layers regardless of their sizes, which would limit the inference performance. In this work, we carry out communication-aware customized optimization for convolutional layers with different sizes. First, we propose a reconfigurable high-performance photonic interconnect (HPPI) architecture to facilitate the communication in chiplet-based DNN accelerators. Second, we propose a customized dataflow as the mapping framework and provide four communication patterns of the photonic interconnect with different ways of spatial mapping. Third, we propose a lightweight back propagation neural network to efficiently select the optimal communication pattern for each convolutional layer. The proposed photonic interconnect can be switched between the four communication patterns to enable communication-aware customized optimization for each convolutional layer in the DNN model. As compared to Simba (a representative chiplet-based accelerator with electronic interconnect), HPPI reduces the execution time by 72.23% on average, while saving the energy consumption by 25.49% on average. As compared to ASCEND (a state-of-the-art chiplet-based accelerator with photonic interconnect), HPPI reduces the execution time by 34.04% on average, while saving the energy consumption by 10.34% on average.**

***Index Terms*—Chiplet, deep neural network (DNN) accelerator, interconnect network, mapping.**

I. Introduction

With the development of deep learning, deep neural networks (DNNs) have been applied to image recognition, object detection, automatic driving, and other fields [1], [2], [3]. Recent DNN models have significantly increased their complexity and size to achieve higher inference accuracy [4], [5], [6]. Traditional computing platforms (GPU, ASIC, etc.) cannot give consideration to both high performance and computational flexibility [7]. On the other hand, the demand for high-performance computing in an era of slowing transistor scaling has motivated advances in multichip-module (MCM) integration [8]. Chiplet-based accelerator designs using MCM have been proposed to implement high-performance DNN inference [8], [9], [10], [11], [12]. However, prior chiplet-based accelerators with metallic interconnect-based Network-on-Chip (NoC) face a problem that the nonuniform latency and bandwidth in intra-chiplet and interchiplet communication would cause significant latency variability across chiplets [8]. The long-distance communication across chiplets would also consume more energy. All of these limitations constitute a great challenge to computing performance, energy efficiency, and system scalability.

As compared to traditional metallic interconnects, photonic interconnects exhibit low delay and low power consumption in long-distance communication [13]. Chiplet-based accelerators with photonic interconnects have been proposed [14], [15], [16]. Data can be transmitted through the waveguide in one hop regardless of the position of source and destination. In addition, the technology of wavelength-division multiplexing (WDM) and space-division multiplexing (SDM) greatly increase the communication bandwidth of photonic interconnects [17]. ASCEND is a state-of-the-art chiplet-based accelerator designed with photonic interconnect [16]. It achieves efficient interchiplet broadcast communication of inputs and weights through a collaborative optimization of dataflow design and photonic multicast communication mechanism. However, considering the great difference in parameter structures among different convolutional layers, the fixed design of broadcast communication adopted by ASCEND would make it difficult to achieve effective performance optimization for all the convolutional layers in a DNN model. How to achieve communication-aware customized optimization for convolutional layers with different sizes is a great challenge.

In this work, we propose high-performance photonic interconnect (HPPI), an HPPI with reconfigurable wavelength assignment schemes to facilitate the communication in chiplet-based DNN accelerators. On this basis, we propose a customized dataflow as the mapping framework, based on which we construct multiple communication patterns along with corresponding wavelength assignment schemes. Furthermore, we propose a mapping optimization strategy that uses a lightweight back propagation neural network to efficiently select the optimal communication pattern for each

Manuscript received 28 April 2023; revised 19 August 2023 and 10 October 2023; accepted 22 October 2023. Date of publication 31 October 2023; date of current version 21 February 2024. This work was supported by the National Natural Science Foundation of China under Grant 62072298. This article was recommended by Associate Editor I. OConnor. *(Corresponding author: Yaoyao Ye.)*

The authors are with the Department of Micro/Nano Electronics, School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: yeyaoyao@sjtu.edu.cn).

Digital Object Identifier 10.1109/TCAD.2023.3328828

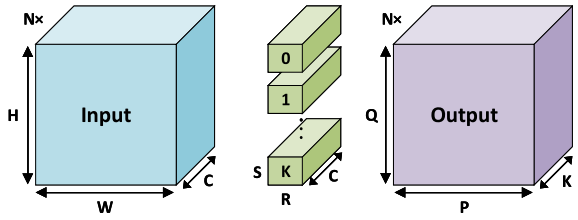

Fig. 1. Overview of a convolutional layer.

```
for n = [0:N):
 for p = [0:P):
  for q = [0:Q):
   for k = [0:K):
    for c = [0:C):
     for r = [0:R):
      for s = [0:S):
       O[n,p,q,k] += I[n,h,w,c] * W[r,s,c,k];
```

Listing 1. DNN loop nest.

convolutional layer in a DNN model. In this way, the proposed photonic interconnect can be switched between the multiple communication patterns to enable communication-aware customized optimization for each convolutional layer in the DNN model.

This article is organized as follows. Section II introduces research background and related works about chiplet-based DNN accelerators. Section III presents the chiplet-based DNN accelerator with the proposed reconfigurable HPPI design with a customized dataflow and a mapping optimization strategy. Section IV introduces the evaluation methodology and the detailed experimental settings. Section V presents the experimental results and comparisons with the baselines. Finally, Section VI concludes this article.

II. Background and Related Work

Applications from edge devices to data centers require fast inference with low latency. However, the performance of the existing chiplet-based DNN accelerators is usually limited by multiply-and-accumulate (MAC) utilization, network bandwidth, and communication latency. The introduction of photonic interconnect into chiplet-based DNN accelerators provides a good solution to these problems. It not only meets the communication requirements of DNN accelerators with low latency and high bandwidth, but also shows a good scalability to different application scenarios.

A. DNN Basics

DNNs are constructed with a series of layers, including convolutional layers, pooling layers, activation layers, and fully connected layers. Generally, in order to reduce data movement, the pooling layers and activation layers are merged with convolutional layers when performing inference tasks [8]. Also, the fully connected layers can be considered as special convolutional layers. Therefore, in this work, the discussion focuses on the convolutional layers.

As shown in Fig. 1 and Listing 1, a convolutional layer is algorithmically formulated as a seven-dimensional nested loop.

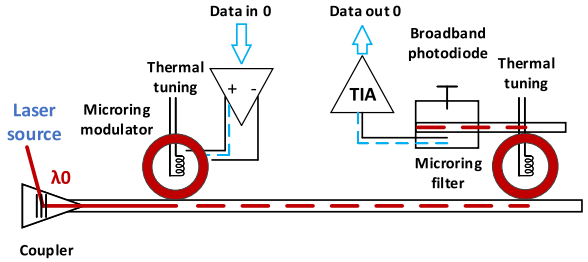

Fig. 2. Basic photonic link.

The seven dimensions include the number of input channels (C), the number of output channels (K), the height (S) and width (R) of weights, the height (Q) and width (P) of output feature maps, and the number of inputs or batch size (N). With reference to the related works Simba [8] and ASCEND [16], the performance evaluation metric is the inference latency with batch size of one. In this work, we also assume the batch size (N) is fixed as one in order to minimize the inference latency in some deployment scenarios like edge device and automatic driving [3]. So, we only need to consider six dimensions.

B. Photonic Interconnects

1) Silicon Photonic Link: Silicon photonics based on-chip communication technology has been studied for more than ten years [18]. Fig. 2 shows an on-chip silicon photonic link composed of microring resonator (MRR)-based modulator and filter. An optical signal emitted by the laser source at the sender would be modulated by the microring modulator, and then be transmitted along the waveguide until to the receiver. The optical signal would be switched by the microring filter and then be converted back into an electrical signal by the receiver. WDM is a technology that enables a significant increase in bandwidth of a photonic link. In a WDM-based photonic link, multiple wavelengths are transmitted in parallel along the same waveguide, and each wavelength is modulated or received individually. Prior work has demonstrated that up to 64 wavelengths can be multiplexed in a single waveguide [19].

2) Multicast and Broadcast Communication: In the photonic network, a tunable splitter is essential for multicast or broadcast. When the splitter is working in the active mode, a fraction (α) of power is transferred to the drop port while the remaining fraction ($1-\alpha$) of power is transferred to the through port. It works by operating a MRR in the transient zone between the full resonance state and the off resonance state for a specific wavelength [20]. The split ratio is defined as $\alpha/(1-\alpha)$, which varies from 0.4 to 1.8 as the bias voltage changes from 0 V to 5 V. If the required split ratio exceeds the above range, multiple splitters in series are required [16].

Fig. 3 shows an example of unicast, multicast, and broadcast communication in a photonic interconnect. As shown in Fig. 3(a), each receiver uses a specific wavelength to realize synchronous unicast communication. Fig. 3(b) shows an example of multicast. The split ratios of the two active tunable splitters are both set to 1. Similarly, Fig. 3(c) shows an example of broadcast. The split ratio of each splitter in this mode

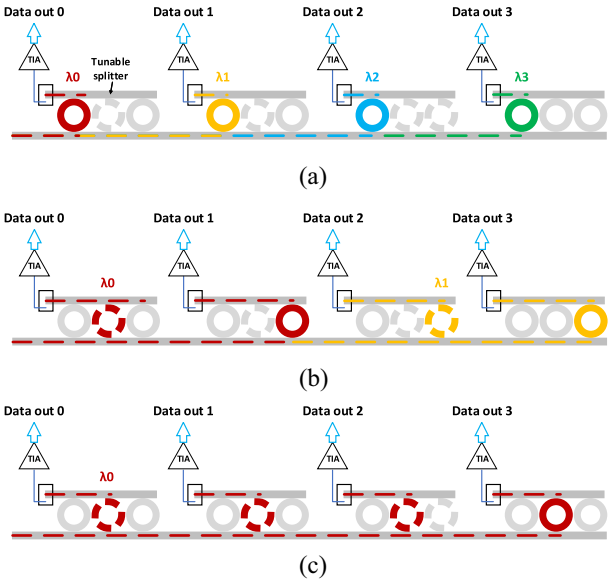

Fig. 3. (a) Unicast, (b) multicast, and (c) broadcast in a photonic link.

are set to 1/3, 1/2, and 1, respectively. Generally, the delay for tuning the optical tunable splitters is set to 500 ps [16]. Thus, it is feasible to achieve a quick switching of communication mode by changing the state of MRRs.

C. Chiplet-Based DNN Accelerators

1) Interconnect Network: For chiplet-based DNN accelerators, the communication efficiency of the interconnect network is essential for the performance and power efficiency of the whole system. Simba is a pioneering chiplet-based DNN accelerator, which uses 2-D mesh-based network to interconnect all the chiplets, and achieves up to 128 TOPs with 32 chiplets in total [8]. Its performance is mainly limited by the long-distance interchiplet communication and the nonuniform communication latency. NN-Baton is another efficient design of chiplet-based DNN accelerator [12]. In NN-Baton, four chiplets are interconnected with a ring-based network, and crossbar is used for on-chiplet communication and memory access. Due to the limitations of the ring topology and the crossbar, its communication architecture would not be suitable for large-scale systems. Besides, Wang et al. [21] proposed a Network-on-Interposer (NoI) design with fat tree topology for DNN accelerators, which is beneficial to broadcast communication. However, its off-package bandwidth would be limited due to that the memory controller is deployed in the center of the package.

On the other hand, in order to take advantage of photonic interconnect to enhance the communication bandwidth and power efficiency, chiplet-based DNN accelerators based on photonic interconnect have been proposed in the literature. Li et al. [14] proposed SPRINT, a chiplet-based accelerator with a single-write multiple-read (SWMR) photonic interconnect. It supports both the unicast and broadcast communication. However, the contention between the unicast of weights and the broadcast of input features would limit the performance. Li et al. [16] further proposed ASCEND, a chiplet-based DNN accelerator with an improved photonic interconnect network that can effectively support the broadcast for both inputs and weights. However, the mismatch between the output feature size and the network structure may significantly reduce the utilization of computing resources.

2) Dataflow and Mapping: As a bridge between DNN inference applications and the accelerator architecture, the dataflow specifies how to divide the DNN workloads and how data can be staged in the memory hierarchy. The ways of scheduling operations and staging data on the same architecture are called mappings [22]. In prior works, Simba and NN-Baton used the weight-stationary (WS) and the output-stationary (OS) dataflow, respectively. In a DNN accelerator, there are lots of inter-PE communication applying the WS dataflow and lots of weight multicast applying the OS dataflow. Wang et al. [21] proposed a weight rotation technology similar to the OS dataflow. Both of SPRINT [14] and ASCEND [16] used customized dataflow to accommodate the communication characteristics of photonic interconnect.

III. High-Performance Photonic Interconnect Design for Chiplet-Based DNN Accelerators

To achieve fast DNN inference in chiplet-based DNN accelerators, we propose an HPPI network that adapts to the communication characteristics of DNN workloads. On this basis, we propose a customized dataflow design with WDM technology as the mapping framework, and provide four communication patterns of the photonic interconnect with different ways of spatial mapping. The proposed photonic interconnect can be reconfigured between different communication patterns to enable communication-aware customized optimization for each convolutional layer in the DNN model. We can determine the optimal communication pattern for each convolutional layer by traversing and comparing the best performance that our design can achieve with each communication pattern. Furthermore, we further propose a lightweight back propagation neural network to effectively select the optimal communication pattern for each convolutional layer.

A. High-Performance Photonic Interconnect Design

Fig. 4 shows an eight-chiplet DNN accelerator architecture with the proposed photonic interconnect network. It can scale up to 32 chiplets. Each chiplet contains 16 processing elements (PE00-PE15). We assume the memory controller is supported by high bandwidth memory 2 (HBM2) technology [23]. The global buffer (GLB) is used to store input features in order to avoid the repetitive accesses to off-chip memory. Regarding the interconnect network, we adopt a similar design as the related works [24], [25]. We assume the photonic interconnection network as well as electrical-to-optical (E/O) and optical-to-electrical (O/E) conversions are implemented on the interposer. The microbump technology is used to connect chiplets to the interposer.

In package level [Fig. 4(a)], chiplets in the same row share three waveguides to obtain input features and weights. The waveguide 0 connected to GLB is responsible for transmitting input features. The remaining two waveguides are directly connected to the memory controller for weights transmission.

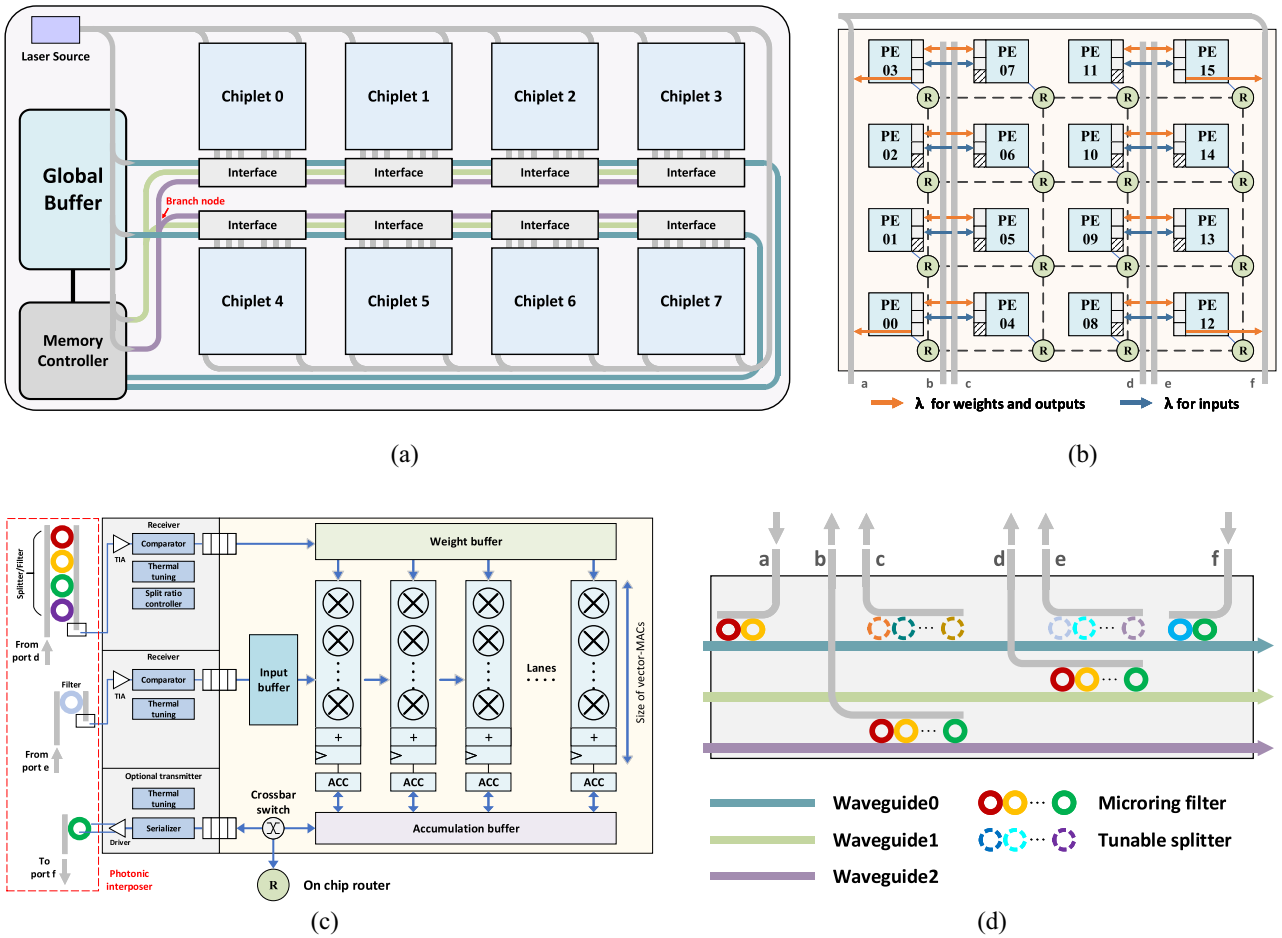

Fig. 4. Overall architecture of an eight-chiplet DNN accelerator with the proposed HPPI. It can scale up to 32 chiplets. (a) Package architecture. (b) Architecture of Chiplet 0. (c) Architecture of PE15 in Chiplet 0. (d) Customized photonic interface.

Each chiplet connects to the three waveguides through a customized photonic interface, which is shown in Fig. 4(d). Ports a, b, and c are connected to PE00-PE07 on each chiplet. Port a is responsible for sending output features from the transmitter in PE00 and PE03 to the memory controller through waveguide 0. Ports b and c are used to send weights and input features to PE00-PE07. Port b is responsible for sending weights from waveguide 2 and port c is responsible for sending input features from waveguide 0. Similarly, ports d, e, and f are connected to PE08-PE15 on each chiplet. The data transmitted through port d, e and f includes weights from waveguide 1, input features from waveguide 0 and output features from the transmitter in PE12 and PE15, respectively. Resonant wavelengths of the microring filter and tunable splitters in the interface will be determined by our wavelength assignment strategy, which will be discussed in Section III-C and III-D.

Additionally, chiplets located in different rows share the same weights. As shown in Fig. 4(a), waveguide 1 connected to the memory controller is divided into two branches, which connect the first and second rows of chiplets, respectively. Signal in each wavelength is divided into two equal parts. The branch node can be implemented by multiple tunable splitters. We adopt the same design for waveguide 2.

On each chiplet [Fig. 4(b)], 16 PEs are divided into two groups according to our photonic interconnect structure: 1) PE00–PE07 are in a group and 2) PE08–PE15 are in another group. PEs in the same group share three waveguides. For example, in the group of PE00-PE07, the waveguides connected to port b and c are used for transmitting data from the interface to each PE, and the other waveguide that connected to port a is used for sending data to the interface. Meanwhile, all the 16 PEs are interconnected to each other by a mesh-based electronic NoC. In the PE level, each PE [Fig. 4(c)] contains eight lanes, where each lane has eight 8-bit precision MAC units. Each PE could achieve up to 64 MACs per cycle. Local buffers in each PE are deployed to store input features, weights, and partial sums.

The number of PEs and MAC units on each chiplet is consistent with Simba's configuration for a fair comparison. The configuration of waveguides is based on the following considerations.

1) Each PE can access data through the photonic interconnection network directly.
2) No more than 64 wavelengths are used in a single waveguide.
3) Connect PEs with the same data requirements to the same waveguide for an efficient multicast.

Furthermore, our design of PEs grouping is based on the following considerations. For each row of chiplets, the number of wavelengths used for transmitting input is usually less than the number of wavelengths used for transmitting weight. We connect different groups of PEs to different interchiplet weight waveguides while sharing the same interchiplet input waveguide.

To enable the optical/electronic conversion, we deployed two receivers and an optional transmitter for each PE. The receiver for weights is connected to four MRRs with different resonant wavelengths. These MRRs are composed of filters or splitters and only one MRR is active while the others are passive. We assume the resonance wavelength of MRR is controlled by adjusting the voltage. When an MRR is active by applying a bias voltage, signal will be coupled into the microring and be directed to another waveguide. Otherwise, when the MRR is passive by powering off, signal will not be coupled into the microring, but will propagate along the input waveguide. Its specific combination form and resonant wavelengths are determined according to the wavelength assignment. Meanwhile, the receiver for inputs is connected with a specific wavelength. These two receivers can work simultaneously without mutual interference to realize high read bandwidth. When the partial sums finish its local accumulation, it is sent to other PEs through the mesh-based electronic NoC for further accumulation. The output features that have completed all the accumulation operations will be sent to the nearest transmitter and then be transmitted back to the memory controller through interchiplet waveguides. Its communication behavior is determined by specific mapping.

We assume that each chiplet has a control unit that modulates the voltage applied to MRRs based on the mapping scheme. The peripheral circuitries of the transmitter and receivers on each chiplet are connected to corresponding photonic components on the photonic interposer through micro-bumps [17].

```
1  // System level
2  for p4 = [0:P4):
3   for q4 = [0:Q4):
4    for k3 = [0:K3):
5  // Package level
6  for p3 = [0:P3):
7   for q3 = [0:Q3):
8    // Split along rows
9    parallel_for p2 = [0:P2):
10    parallel_for q2 = [0:Q2):
11     // Split along columns
12     parallel_for k2 = [0:K2):
13 // Chiplet level
14 parallel_for p1 = [0:P1):
15  parallel_for q1 = [0:Q1):
16   parallel_for c2 = [0:C2):
17 // PE level
18 for k1 = [0:K1):
19  for c1 = [0:C1):
20   for r = [0:R):
21    for s = [0:S):
22     parallel_for k0 = [0:K0):
23      parallel_for c0 = [0:C0):
24 // MAC level
25 p = ((p4 * P3 + p3) * P2 + p2) * P1 + p1;
26 q = ((q4 * Q3 + q3) * Q2 + q2) * Q1 + q1;
27 k = ((k3 * K2 + k2) * K1 + k1) * K0 + k0;
28 c = (c2 * C1 + c1) * C0 + c0;
29 O[p,q,k] += I[p-1+r,q-1+s,c] * W[r,s,c,k];
```

Listing 2. Proposed customized dataflow for HPPI.

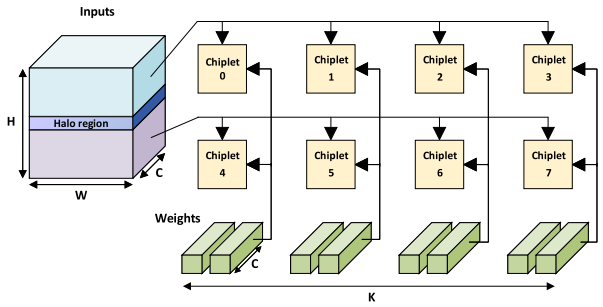

Fig. 5. Example of spatial mapping across chiplets.

B. HPPI Dataflow

We propose a customized dataflow (Listing 2) as the mapping framework that cooperates with the proposed interconnect network. Different mapping schemes based on this dataflow will correspond to different wavelength assignments in the proposed photonic network. And also, different wavelength assignments provide different network bandwidth available. We provide four communication patterns for the proposed photonic interconnect. Each communication pattern corresponds to a different wavelength assignment scheme. Furthermore, we propose a mapping optimization strategy that selects the optimal communication pattern for each convolutional layer. By reconfiguring the proposed photonic interconnect to work in the optimal communication pattern for each layer, the performance of the DNN accelerator can be maximized.

Listing 2 shows the detailed structure of the proposed HPPI dataflow. At the system level, the dataflow uses an output centric loop tiling (lines 2–4) in the off-package memory to ensure that all the buffers can meet the requirements of different convolutional layers. Otherwise, there may be situations where no valid mappings can be found. At the package level, it describes the loop tiling in GLB (lines 6 and 7) and the spatial division across chiplets (lines 8–12), which means that $P2 \times Q2$ rows and $K2$ columns of chiplets are involved in the computation. The convolutional layer is divided into subworkloads on these dimensions. All the subworkloads are sequentially mapped on each chiplet in a loop order. Fig. 5 shows an example of spatial mapping at package level, which describes the case of $P2 = 1$, $Q2 = 2$, $K2 = 4$. Chiplets in the same row share the same input features, and chiplets in the same column share the same weights. The halo region represents the repeated GLB accesses caused by the overlap of convolution windows which will result in additional energy consumption. Thus, at package level, we only allow P-dimension and Q-dimension divisions along rows to constrain this additional energy consumption. Significantly, considering the high cost of interchiplet communication, we remove the spatial partitioning of input channels (C) at this level and avoid the accumulation traffic between chiplets. At the chiplet level, it shows a special spatial division (lines 14–16) and uses $P1 \times Q1 \times C2$ PEs per chiplet.

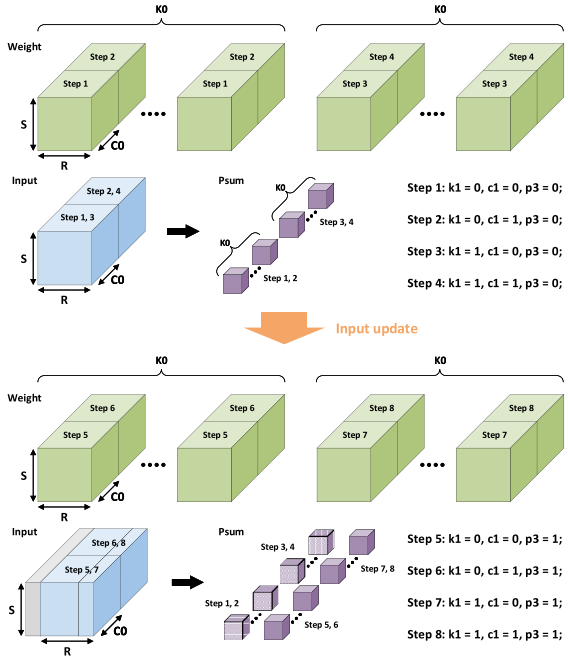

Fig. 6. Example of processing steps in one PE.

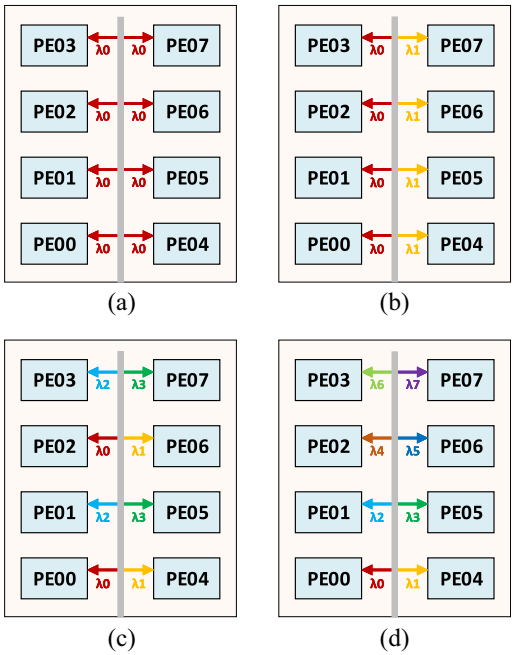

Fig. 7. On-chip wavelength assignment for weights transmission in different communication patterns. (a) Pattern 0. (b) Pattern 1. (c) Pattern 2. (d) Pattern 3.

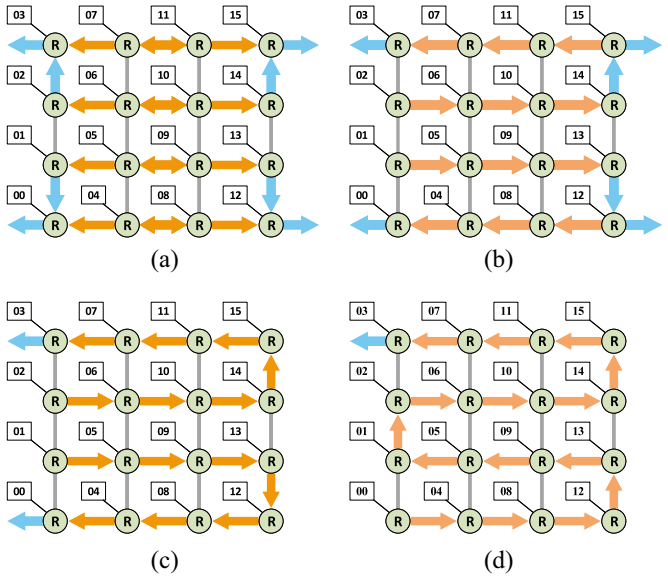

Fig. 8. On-chip traffic in different communication patterns. (a) Pattern 0. (b) Pattern 1. (c) Pattern 2. (d) Pattern 3.

In prior output stationary dataflow, the accumulation of partial sums between PEs is always prohibited in order to reduce the communication latency. However, we note that in some cases, splitting input channels across PEs is beneficial to reduce the total execution delay of the DNN model.

At the PE level, the loop tiling (line 18–21) focuses on data reuse to avoid bandwidth constraints and improve the energy efficiency. Fig. 6 shows an example of the detailed processing steps in one PE. In this example, for brevity, we assume that $K1 = C1 = 2$. All these data blocks are temporarily stored in PE's local buffer. Each step represents the multiplication and accumulation of the corresponding data blocks. The result of multiplication is accumulated to the partial sums stored in the accumulation buffer. After all the results from steps 1–4 are accumulated, these partial sums will be sent to other PEs through the mesh-based electronic NoC for further accumulation. Steps 5–8 represent another round of multiplication and accumulation operation after updating the inputs. Lines 22 and 23 of Listing 2 show the spatial partitioning across lanes in PE and MACs in lane. Once the mapping scheme is generated offline, the interconnect network and computing units execute the computing tasks specified in the mapping scheme in a pipeline manner [22].

C. On-Chiplet Wavelength Assignment

In this section, we will introduce four on-chiplet communication patterns associated with specific wavelength assignments. Different communication patterns represent different ways of spatial mapping and provide a foundation for building a highly flexible dataflow. As shown in Fig. 4(b), in each group, multiple wavelengths are coupled into the waveguide for receiving data, which will be used for the unicast of input features and the multicast of weights. The weights received by two groups on the same chiplet come from different inter-chiplet photonic links. The two groups can apply the same wavelengths for weights transmission without interfering with each other. It helps save the number of wavelengths occupied by each chiplet, thereby limiting the number of wavelengths used by the entire system to a reasonable range. In addition, four transmitters are deployed in four corners of a chiplet. These transmitters use four different wavelengths to write the output features back to off-chip memory.

As shown in Figs. 7 and 8, we classify the on-chiplet communication as four communication patterns. The total number of communication patterns can be calculated as $\log_2 N + 1$,

where N represents the number of PEs per waveguide. In our work, eight PEs in each group obtain weight data through the same waveguide, so there are four communication patterns. Different communication patterns have different number of multicast destinations. To be specific, in pattern 0, the number of multicast destinations is eight; in pattern 1, the number of multicast destinations decreases to four; in pattern 2, the number of multicast destinations further decreases to two; and in pattern 3, the number of multicast destinations is one, which is equal to a unicast communication.

For brevity, here we only show the detailed wavelength assignments related to weights in one group of Chiplet 0. The input-related wavelength assignment in the four communication patterns are the same, that is, wavelengths λ32–λ39 are used to transmit input features to these eight PEs separately. The details of the four communication patterns are as follows.

1) *Communication Pattern 0:* Fig. 7(a) shows the wavelength assignment in Pattern 0, where PE00–PE07 share weights through wavelength λ0. It corresponds to the case that $P1 \times Q1 = 8$ and $C2 = 2$ in lines 14–16 of the HPPI dataflow (Listing 2). Fig. 8(a) shows the on-chip communication traffic that is involved in this pattern. The partial sums generated by the third column are sent to the first column, and the partial sums generated by the second column are sent to the fourth column. When the output features finish its accumulation, it will be written back to the memory controller through the nearest transmitter. For example, the outputs generated in PE00 and PE01 will share the optical transmitter in PE00 and the other PEs will act in the same rules.
2) *Communication Pattern 1:* Fig. 7(b) shows the wavelength assignment in Pattern 1, where PE00–PE03 share the same weights through wavelength λ0 and PE04–PE07 share the same weights through wavelength λ1. It corresponds to the case that $P1 \times Q1 = 4$ and $C2 = 4$ in lines 14–16 of the HPPI dataflow. Partial sums from four PEs are accumulated sequentially through the mesh-based electronic NoC. Fig. 8(b) shows an example of on-chip communication traffic in this pattern. Similar to Pattern 0, output features generated in PE13 and PE14 are written back to the memory controller through the nearest optical transmitters in PE12 and PE15, respectively. The output features generated in PE00 and PE03 will be transmitted through local transmitters.
3) *Communication Pattern 2:* Fig. 7(c) shows the wavelength assignment in Pattern 2, where wavelengths λ0-λ3 are used for the multicast of weights. It corresponds to the case that $P1 \times Q1 = 2$ and $C2 = 8$ in lines 14–16 of HPPI dataflow. As shown in Fig. 8(c), due to the high cost of transmitting output features between PEs, output features that have completed eight accumulations are written back to the memory controller through the optical transmitters in PE00 and PE03. Only half of the optical transmitters are used in this pattern.
4) *Communication Pattern 3:* Fig. 7(d) shows the wavelength assignment in Pattern 3, where eight wavelengths (λ0-λ7) are used to transmit weights to PE00–PE07, respectively. It corresponds to the case that $P1 \times Q1 = 1$ and $C2 = 16$ in lines 14–16 of HPPI dataflow. Partial sums from all the sixteen PEs need to be accumulated, as shown in Fig. 8(d). Once the output features finish all 16 accumulations in PE03, it will be transmitted by the optical transmitter in PE03. Attention that only one optical transmitter is used.

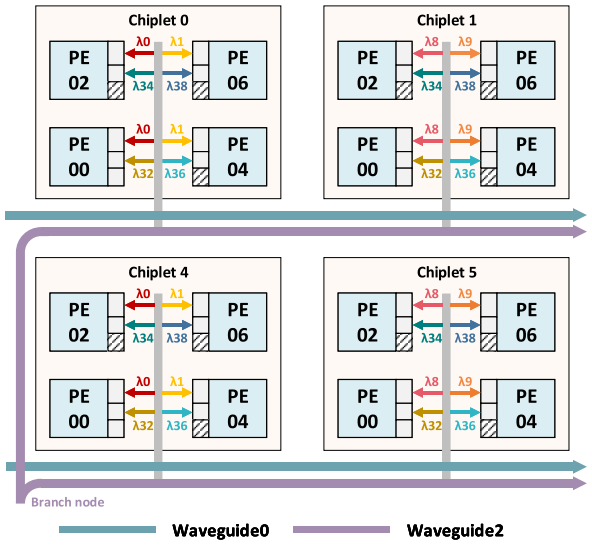

Fig. 9. Interchiplet communication in Pattern 2.

D. Interchiplet Wavelength Assignment

In this section, we will propose a wavelength assignment strategy for the multicast and unicast between chiplets.

Weight Multicast: According to lines 8–10 of the proposed customized dataflow for HPPI (Listing 2), chiplets located in the same column have the same demand for weights. Fig. 9 shows an example of Pattern 2 that PE00 and PE02 in Chiplet 0 and Chiplet 4 share the same weights through wavelength λ0. The remaining three columns of PEs also share weights in the same way. As described in Section III-C, each group uses up to eight wavelengths for weights transmission to support Pattern 0-3. Further more, according to lines 11 and 12 of HPPI dataflow (Listing 2), chiplets located in different columns have different demands for weights. Thus, wavelengths λ0–λ7 are assigned to the first column of chiplets (Chiplet 0 and Chiplet 4). Wavelengths λ8–λ15 are assigned to the second column of chiplets (Chiplet 1 and Chiplet 5). In this way, the maximum number of wavelengths that used to weights transmission is 32 (λ0–λ31). If the accelerator system is expanded to eight chiplets per row, the maximum number of wavelengths used for weight transmission will increase to 64 (λ0–λ63).

Input Multicast: The multicast communication from the GLB to all chiplets in the same row takes place in the same waveguide. For example, all the PE00 in Chiplets 0–3 share the same input features through wavelength λ32 in an eight-chiplet accelerator. Such multicast is directed toward each PE on the same row of chiplets. And also, multicast between chiplets in each row is independent. The waveguides that provide input features for each row of chiplets access the data in GLB through their own independent data interfaces. Considering

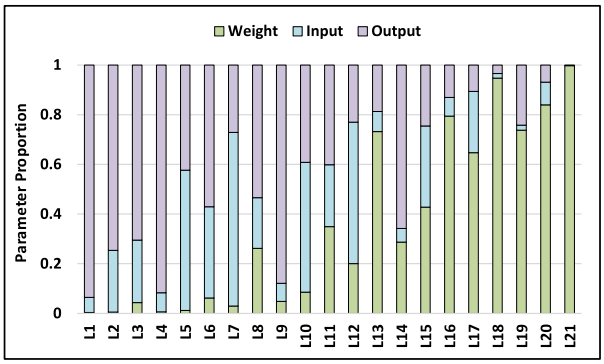

Fig. 10. Parameters proportion of ResNet-50.

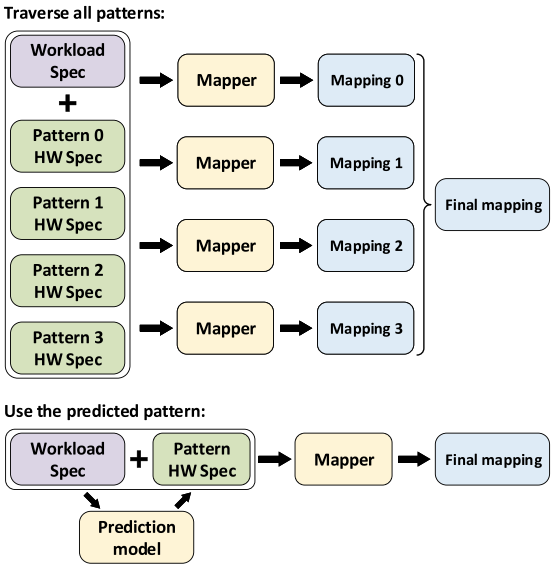

Fig. 11. Execution process of two pattern selection methods.

that each PE on the chip requires a separate wavelength, the total number of wavelengths for input features in the waveguide is 16 (λ32–λ47). If λ32–λ47 have been assigned to weight transmission, the wavelengths used for input transmission are postponed to λ64–λ79.

Output Unicast: To provide sufficient write bandwidth, we deploy optical transmitters on PE00, PE03, PE12 and PE15 of each chiplet. These transmitters reuse the wavelengths that originally allocated for weights multicast. For example, in Chiplet 0, λ0–λ3 are allocated to PE00, PE03, PE12, PE15, respectively, [Fig. 4(b)]. And also, λ8–λ11 are used for transmitting accumulated output features from Chiplet 1 to memory controller. All these wavelengths share the waveguide with those wavelengths that are allocated to transmit inputs. Such assignment ensures that inputs and outputs communication can share the same waveguide while remaining undisturbed.

E. Fast Optimal Pattern Prediction Model

In order to choose the optimal communication pattern for each convolutional layer, the basic approach is to go through every combination of communication patterns for all the layers, and select a globally optimal mapping with the best performance as the final deployment scheme. We call this traversal method. However, this method leads to the expansion of search space and thus would inevitably increase the time the mapper takes to map the DNN model to the accelerator.

To alleviate this problem, we have trained a lightweight back propagation neural network as a fast pattern prediction method to skip the traversal process and directly recommend a communication pattern for each layer. Fig. 10 shows the proportion of three kinds of parameters in 21 different layers of ResNet-50 [1]. The overall trend is that the ratio of weight increases gradually from Layer 1 (L1) to Layer 21 (L21), while the ratios of input and output decrease gradually. In subsequent experiments, we noticed a clear correlation between parameter proportion and the optimal choice of communication pattern. Thus, we use a back propagation neural network to build the pattern prediction model, which is able to predict the optimal communication pattern for each layer. On the other hand, for specific hardware parameters, we may find a specific formula to express the regression relationship between parameter characteristics and pattern selection. However, if the computing capability or link bandwidth changes, the regression relationship will also change. To express the new regression relationship, it is much easier to retrain the prediction model than to find a new specific formula. Thus, we choose the prediction model rather than a specific formula.

The model is trained by the parameter proportion and the optimal communication patterns of multiple layers from ResNet-50 [1] and GoogLeNet [26]. ResNet-50 is a representative model of the deep residual neural networks that are widely used nowadays. The diversity of its layers provides sufficient breadth to cover most convolutional layers and fully connected layers in deep residual neural networks. In addition, GoogLeNet is composed of multiple inception modules, which contain more convolutional kernels with different sizes. We conduct a detailed performance analysis of ResNet-50 with four patterns in order to select the optimal communication pattern for each layer. Similarly, we use the traversal method to select the optimal communication pattern for each layer in GoogLeNet. And then, we build the proposed pattern prediction model based on the communication pattern selection results of ResNet-50 and GoogLeNet. Once the model training is complete, it can directly dock with the target DNN model that needs to be executed on the accelerator.

As shown in Fig. 11, we compare the execution processes of the two methods mentioned above. The traversal method needs to call the mapper of Timeloop [22] in each communication pattern which have different hardware parameters. The proposed prediction model simplifies this process. After the prediction model provides a communication pattern, the Timeloop's mapper then performs mapping search in this specific pattern. Our prediction model is used offline. The prediction delay is negligible compared with the runtime of the Timeloop's mapper. In Section V, we will apply the trained pattern prediction model to the other four DNN models (ResNet-18 [1], VGG-16 [27], Darknet-19 [28], and AlexNet [29]) and compare their inference performance with other baseline accelerator designs.

IV. Evaluation Methodology

In this section, we will introduce the experimental methods in detail including accelerator architecture parameters, performance evaluation model and energy consumption evaluation model. We choose several mainstream DNN models as the benchmarks for analysis of performance and energy consumption, including ResNet-18, ResNet-50, VGG-16, Darknet-19, AlexNet, and GoogLeNet. For the sake of discussion, we merge convolutional layers with the same architecture parameters. As a result, there are 12, 21, 12 and 11 different layers in ResNet-18, ResNet-50, VGG-16, and DarkNet-19, respectively.

A. Accelerator Architecture Parameter Configuration

For experimental analysis and comparisons, we assume the proposed HPPI accelerator is with 32 chiplets [arranged in four rows and eight columns as in Fig. 4(a)]. The main memory type is HBM2. According to work [23], the size of main memory is configured as 4 GB and its access bandwidth can reach up to 256 GB/s. Each chiplet contains 16 PEs, where each PE has 64 MAC units. The total number of MAC units in the proposed HPPI accelerator is 32 768. It adopts the proposed wavelength assignment scheme as described in Section III. Up to 64 wavelengths ($\lambda 0$–$\lambda 63$) are assigned for weight transmission and 16 wavelengths ($\lambda 64$–$\lambda 79$) for input transmission.

To demonstrate the advantages of the proposed HPPI accelerator, we choose Simba [8] and ASCEND [16] as the baseline accelerator designs for comparisons. Simba is a pioneering work in the field of multichiplet DNN accelerators with only metallic interconnects. It has complete and advanced architecture design, so it has important reference significance to compare with Simba. To the best of our knowledge, ASCEND is the state-of-art chiplet-based DNN accelerator with photonic interconnects. For fair comparison, we assume the same computing resources (32768 MACs) for Simba and ASCEND. We assume that both the Simba and ASCEND have 32 chiplets and 1024 MAC units per chiplet. Simba contains 16 PEs per chiplet and 64 MAC units per PE, while ASCEND contains 32 PEs per chiplet and 32 MAC units per PE. Similar to work [8], the local buffer size of each PE is uniformly set to 43 kB. The GLB size of the proposed HPPI accelerator and ASCEND are both set to 2 MB, while the GLB size of Simba is set to 64 kB per chiplet. Besides, we assume that PEs and interconnect network operate at 1 GHz.

Table I shows the network parameters of these three accelerators. For both HPPI and Simba, we assume the read bandwidth is 320 Gb/s per chiplet. The proposed HPPI uses up to 80 wavelengths, thus it achieves 1280-Gb/s weights access from off-package memory. To make a fair comparison, we assume that ASCEND uses 20-Gb/s bandwidth for weights transmission and 10-Gb/s bandwidth for inputs transmission. Hence, we assume ASCEND's read bandwidth is 30 Gb/s per PE, and its read bandwidth is 660 Gb/s per chiplet according to its interconnect network structure.

TABLE I
Network Parameters

Simba	Chiplet level	Electronic mesh 20 Gbps / PE read / write bandwidth
	Package level	Electronic mesh 320 Gbps / chiplet read / write bandwidth
ASCEND	Chiplet level	Photonic interconnect 30 Gbps / PE read bandwidth 40 Gbps / PE write bandwidth (share)
	Package level	Photonic interconnect 660 Gbps / chiplet read bandwidth 40 Gbps / chiplet write bandwidth
HPPI	Chiplet level	Electronic mesh 20 Gbps / PE read bandwidth 20 Gbps / metallic link
	Package level	Photonic interconnect 320 Gbps / chiplet read bandwidth 10-40 Gbps / chiplet write bandwidth 10 Gbps $\times$ 24-80 wavelengths

B. Simulation Platform

We use the open-source Timeloop simulator [22] to evaluate the performance and energy consumption of HPPI and the two baseline DNN accelerator designs Simba and ASCEND. For performance evaluation, the Timeloop simulator supports the nonuniform bandwidth between chiplets and PEs. Further more, in a multichiplet system, the accumulation delay among PEs or chiplets is an important component of the total execution time of a DNN model. We refer to the theory and method of previous works [30], [31], [32] to estimate the intrachiplet and interchiplet accumulation delay. In the process of communication pattern switching, it is necessary to tune the power of laser source. Referring to work [33], we assume that the power tuning delay of the laser source is 500 cycles. In our design, only the power of the wavelengths responsible for transmission weight needs to be tuned. After all weights are sent to PEs, laser power tuning can be started. The power tuning is completed before all calculations of the current layer are completed. In other words, the laser power tuning delay is covered by the current layer's execution time, and the delay caused by pattern switching can be ignored. Meanwhile, we extend the Timeloop simulator to simulate the behavior of interchiplet photonic interface

$$P_{\text{total}} = P_{\text{compute}} + P_{\text{access}} + P_{\text{photonic}} + P_{\text{electronic}}. \quad (1)$$

As shown in (1), the overall power consumption P_{total} consists of four parts: 1) computation power generated by MAC units (P_{compute}); 2) power for memory hierarchy access (P_{access}); 3) power for photonic interconnects (P_{photonic}); and 4) power for electronic interconnects ($P_{\text{electronic}}$). According to the above equation, we use Accelergy [34] integrated in Timeloop to estimate energy consumption at the architecture level. All the baseline designs and our design utilize 8 bits quantization for weight and input and 24 bits for partial sum. The energy breakdown of metallic-based interconnects is 1.17 pJ/bit for interchiplet communication [35] and 0.0711 pJ/bit for intrachiplet communication [31]. The power estimation of photonic interconnects is similarly to

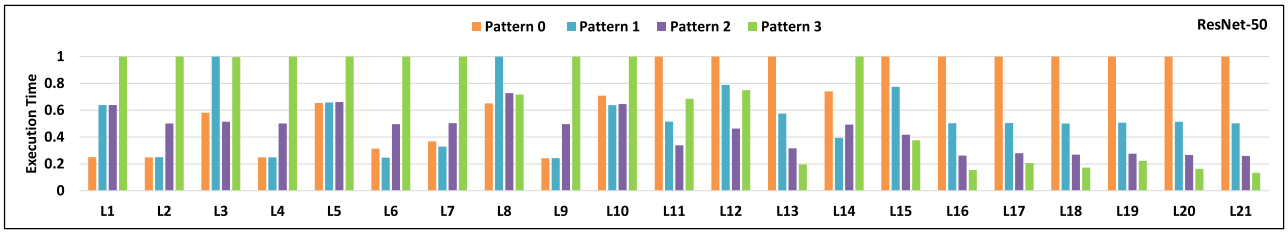

Fig. 12. Normalized execution time of ResNet-50 with four patterns of HPPI.

TABLE II
PHOTONIC PARAMETERS [16]

Component	Value	Component	Value
Laser source	5 dB	Ring drop	1 dB
Coupler	1 dB	Ring through	0.02 dB
Splitter	0.2 dB	Photo-detector	0.1 dB
Waveguide	1 dB/cm	Waveguide-to-receiver	0.5 dB
Waveguide bend	1 dB	Receiver sensitivity	-20 dBm
Waveguide crossover	0.05 dB	Ring heating	2 mW

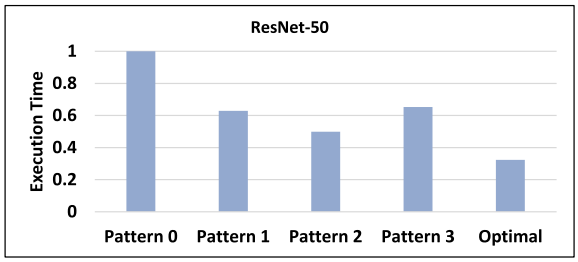

Fig. 13. Normalized execution time with static/optimal pattern configuration.

ASCEND. Important parameters of photonic interconnect are shown in Table II.

Previous studies [12], [36] have shown that the performance of DNN accelerators is sensitive to mapping. For fair comparison, we use the random-pruned search algorithm integrated in Timeloop to find the optimal mapping for a given pattern configuration. The mapper is set to prioritize mappings with the minimum execution time. If two mappings have the same delay, it prioritizes the mapping with lower energy consumption.

V. RESULT COMPARISON AND ANALYSIS

In this section, first, we use ResNet-50 as an example to conduct a case study about the impact of pattern selection on the inference performance. Second, we compare the proposed HPPI accelerator with Simba [8] and ASCEND [16] layer by layer in terms of performance and energy consumption. Third, the prediction accuracy of the proposed pattern prediction model is evaluated. Finally, we compare our design with Simba and ASCEND in terms of the execution time of one complete inference.

A. Impact of Pattern Selection on Inference Performance

In our multipattern design, the accelerator can be reconfigured to work in four communication patterns. Different communication patterns correspond to different configurations of network bandwidth available. Pattern 0 has the highest output write bandwidth and the lowest weight read bandwidth, which is better for convolutional layers with a larger proportion of output. Pattern 3 has the highest weight read bandwidth and the lowest output write bandwidth, which is more suitable for convolutional layers that require more weight accesses from off-chip memory. Patterns 1 and 2 provide a compromise solution as transitional patterns.

To analyze the impact of pattern selection on HPPI's performance, we take ResNet-50 as an example and compare the execution time of each layer (L1–L21) with four communication patterns of HPPI in Fig. 12. The execution time of each layer is normalized to the maximum execution time (e.g., for L1, the Pattern 3 corresponds to the maximum execution time). If we divide all these layers into two parts (L1–L10, L11–L21), we can observe that the trend of execution time with the communication pattern is completely opposite in these two parts. At the same time, layers near L11 (such as L10, L12, and L14) achieve the best performance with Pattern 1 or 2. Generally speaking, from L1 to L21, the optimal pattern for each layer gradually changes from Pattern 0 to Pattern 3. This phenomenon comes from the different parameter structures of different convolutional layers. For example, when executing layers with a large proportion of weight, performance bottlenecks are more likely to occur on links that are responsible for transmitting weights. The performance bottleneck of layers with a large proportion of output appears more in the links for transmitting the output features and partial sums. Our multipattern design provides solutions to alleviating these performance bottlenecks layer by layer. Thus, because of greater flexibility and universality, our design can achieve higher DNN inference performance than those with fixed interconnect networks. Fig. 13 shows the total execution time of HPPI with each single pattern, respectively. We compare them with the execution time of HPPI with the optimal pattern configuration. It shows that our design achieves better performance than static pattern configuration.

On the other hand, our experimental results show that the energy consumption of HPPI varies little when using different communication patterns. The primary energy consumption of the accelerator comes from accessing memory hierarchies. The four communication patterns constrained by the customized dataflow avoid redundant accesses to GLB and memory as much as possible. Therefore, the difference in energy consumption for accessing memory hierarchies is small. As for communication energy consumption, when the total number

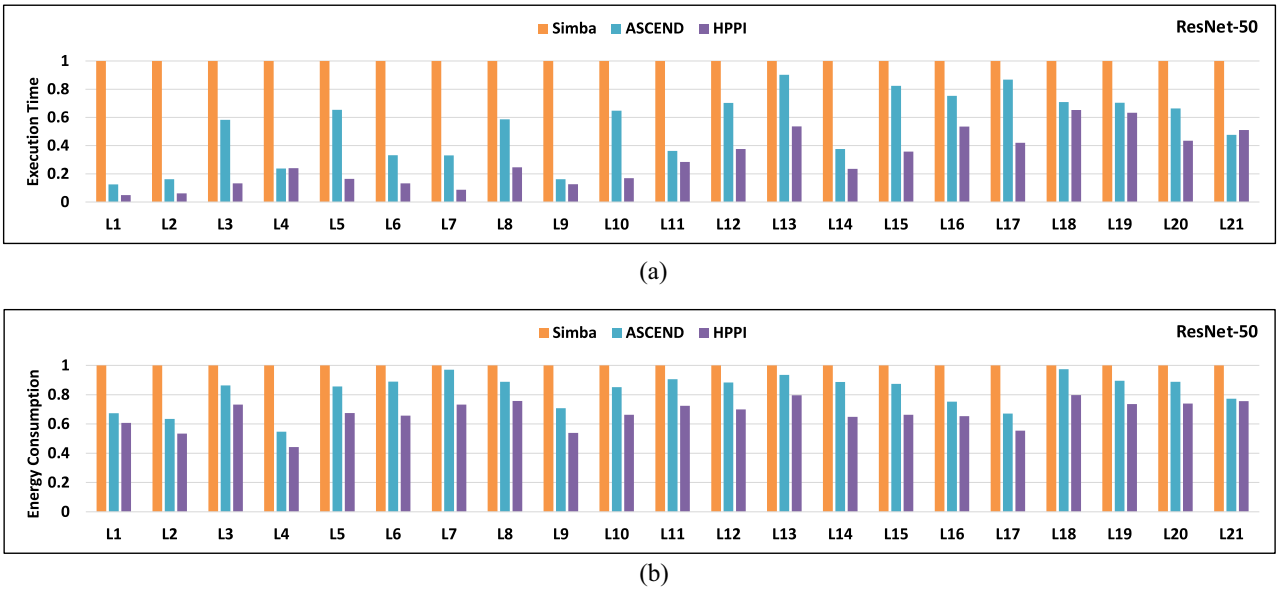

Fig. 14. Layer by layer comparison with Simba and ASCEND. (a) Execution time comparison of ResNet-50. (b) Energy consumption comparison of ResNet-50.

of wavelengths increases, the laser power per wavelength decreases. Overall, the energy consumption differences among the four patterns are minimal. Therefore, the comparison of energy consumption between different communication patterns is not provided in detail.

B. Layer-by-Layer Comparisons for Performance and Energy Consumption

In this section, we take ResNet-50 as an example to compare the proposed HPPI accelerator with Simba [8] and ASCEND [16] layer by layer in terms of performance and energy consumption. Fig. 14(a) shows the comparison for execution time of ResNet-50 in a layer by layer manner. Here, the HPPI has adopted the optimal communication pattern for each layer with the minimum execution time in Fig. 12. All the execution times are normalized to Simba. As compared to Simba, HPPI achieves 35%–95% reduction in execution time. The difference in optimization effects for different layers mainly comes from the following aspects: the utilization rate of PEs, the communication delay, and the difference in data reuse. As compared to ASCEND, HPPI achieves 8%–77% execution time reduction in layers except for L4 and L21. L21 is actually a fully connected layer of ResNet-50. The execution time of L21 is mainly determined by the maximum bandwidth of the transmission weights, which is configured as the same in the two accelerators. In addition, the accumulation delay in our design is inevitable, which brings a small amount of communication delay. Thus, the L21 execution time of HPPI is slightly longer than that of ASCEND. The performance improvement of HPPI over ASCEND mainly comes from our multipattern design which makes full use of the multicast mechanism and network bandwidth available.

Fig. 14(b) shows the comparisons for energy consumption of ResNet-50 in a layer by layer manner. The data listed here is the energy consumption corresponding to the communication pattern with the minimum execution time in Fig. 12. All the data are normalized to Simba. As compared to Simba, HPPI achieves 18%–56% saving of energy consumption, which is achieved by the better energy efficiency of photonic interconnects over electronic interconnects. As compared to ASCEND, HPPI achieves 2%–27% energy reduction by its better data locality.

C. Prediction Model Analysis

Taking ResNet-18 as an example, our model has mispredictions in the four layers L3–L5 and L9. Fig. 15(a) shows the above layers' execution time with different communication patterns, respectively. Take L4 as an example. By the traversal method, we can find that Pattern 0 is the real optimal communication pattern. But the predicted result is Pattern 1. In this case, there is some performance loss due to the prediction bias. From Fig. 15(a), we can observe that only L4 and L9 experience obvious performance loss. In comprehensive consideration of the whole DNN execution process, such performance loss due to the prediction bias will be diluted by other layers that are accurately predicted. Therefore, the accuracy of the proposed prediction model should also be evaluated in terms of the overall performance loss caused by the prediction bias.

To facilitate discussion, we define the prediction accuracy as the overall execution time of a DNN model with the real optimal pattern divided by the execution time with the predicted optimal pattern. This indicator also characterizes the inference speed of the accelerator when executing the DNN model. As shown in Fig. 15(b), the inference speed of ResNet-18 using the pattern prediction method decreases by only 1.95% compared to using the traversal method of pattern selection. We also apply the proposed pattern prediction model to multiple DNN models. The inference speed of DarkNet-19,

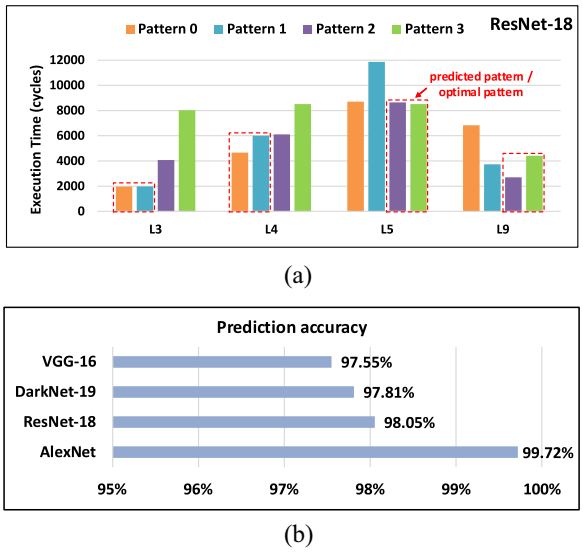

Fig. 15. Prediction model analysis. (a) Performance loss due to prediction bias for four layers of ResNet-18. (b) Accuracy of the pattern prediction for the overall execution time.

VGG-16 and AlexNet decreases by only 2.19%, 2.45%, and 0.18%, respectively.

D. Comparisons for Execution Time and Energy Consumption for One Complete Inference

As shown in Fig. 16, we have made a detailed comparison for execution time and energy consumption of one complete ResNet-50 inference. HPPI has a higher PE utilization and therefore a lower computing latency. The customized dataflow enables better data reuse and thus reduces the additional delay caused by bandwidth limitation. HPPI has a higher network latency than ASCEND, which is the latency caused by the accumulations between PEs. In overall, we have achieved a better performance than the two baselines. As for energy consumption, the DRAM and SRAM access energy consumption accounts for the majority of the total energy consumption in all designs. The advantages of HPPI are reflected in the reduction of SRAM access energy consumption. This is mainly achieved by avoiding repeated access to GLB. In addition, the efficient photonic interconnect networks have also contributed to reducing the total energy consumption.

Table III shows the absolute values of execution time and energy consumption statistics of six mainstream DNN models with different accelerator designs (batch size is one), where HPPI-pre is short for HPPI-predict and HPPI-tra is short for HPPI-traversal. Fig. 17 shows the normalized comparisons with Simba and ASCEND, where all values are normalized to Simba. For ResNet-18, VGG-16, DarkNet-19, and AlexNet, we present both the results of HPPI with the real optimal pattern selection (HPPI-traversal) or with the predicted optimal pattern selection (HPPI-predict). As compared to the real optimal pattern selection, the predicted optimal pattern selection achieves more efficient load deployment at the cost of an average 1.72% increase in the execution time of one complete inference. Since ResNet-50 and GoogleNet have been used to train the pattern prediction model, here our comparisons exclude the HPPI-predict for these two models.

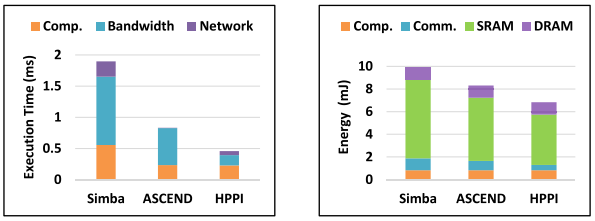

Fig. 16. Breakdown of (a) execution time and (b) energy consumption for one complete ResNet-50 inference.

TABLE III
EXECUTION TIME AND ENERGY CONSUMPTION

	Simba	ASCEND	HPPI-pre	HPPI-tra
ResNet-50	1.896 ms 9.898 mJ	0.831 ms 8.314 mJ	0.457 ms 6.831 mJ	0.457 ms 6.829 mJ
GoogLeNet	1.317 ms 3.845 mJ	0.406 ms 3.097 mJ	0.227 ms 2.799 mJ	0.227 ms 2.798 mJ
ResNet-18	1.166 ms 4.099 mJ	0.391 ms 3.543 mJ	0.179 ms 3.131 mJ	0.176 ms 3.102 mJ
DarkNet-19	1.230 ms 6.496 mJ	0.437 ms 5.604 mJ	0.310 ms 4.835 mJ	0.303 ms 4.833 mJ
VGG-16	3.287 ms 35.450 mJ	1.904 ms 29.670 mJ	1.446 ms 26.787 mJ	1.410 ms 26.580 mJ
AlexNet	1.124 ms 4.988 mJ	0.497 ms 3.916 mJ	0.458 ms 3.937 mJ	0.457 ms 3.937 mJ

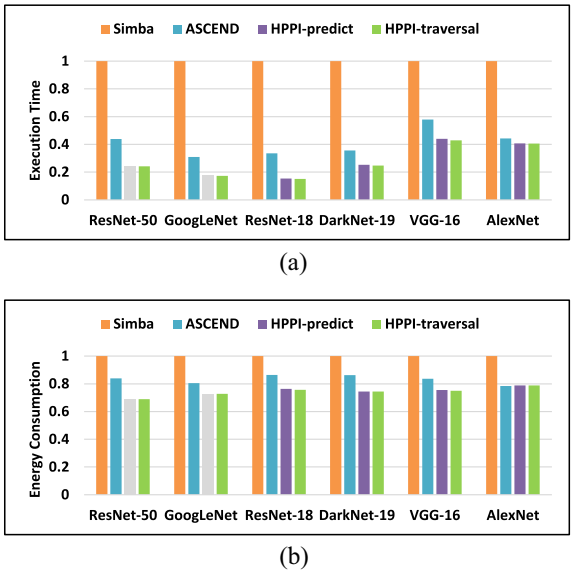

Fig. 17. (a) Normalized execution time and (b) energy consumption for one complete inference.

Fig. 17(a) shows the comparison in terms of the execution time required to perform a complete inference. As compared to Simba, HPPI achieves average execution time reduction by 72.57% with the real optimal pattern selection and 72.23% reduction with the predicted optimal pattern selection for these

DNN models. As compared to ASCEND, HPPI achieves average execution time reduction by 34.80% with the real optimal pattern selection and 34.04% reduction with the predicted optimal pattern selection. On the other hand, Fig. 17(b) shows the comparisons in terms of the energy consumption required to achieve the above inference performance. As compared to Simba, HPPI achieves average energy consumption reduction by 25.70% with the real optimal pattern selection and 25.49% reduction with the pattern prediction method. As compared to ASCEND, HPPI achieves average energy consumption reduction by 10.59% with the real optimal pattern selection and 10.34% reduction with the pattern prediction method.

E. Area Estimation

The chiplet scale of the proposed accelerator (1024 MAC units) is consistent with the chiplet scale of Simba and ASCEND for a fair comparison. According to the requirements of the actual application scenarios, the computing capacity of each PE can be further increased. The proposed interconnect network can still work if the chiplet scaled up beyond 1024 MAC units. The target technology is 16 nm, which is the same as Simba. According to data provided by Simba, the area of each PE excluding the transmitter and receivers is 0.107 mm^2. The total area of PEs on each chiplet is 1.712 mm^2. The area for a transmitter or a receiver is assumed to be 0.0096 mm^2 according to work [37] and their total area is 0.3456 mm^2. Furthermore, we assume microbump pitch size is 36 μm and MRR radius is 5 μm as same as ASCEND. Since most MRRs and microbumps are implemented underneath chiplets, we assume that they will not incur additional area overhead.

VI. Conclusion

In this article, we propose HPPI, an HPPI design for chiplet-based DNN accelerators. The innovative features of HPPI include: 1) a reconfigurable photonic interconnect network that supports multiple communication patterns with diversified wavelength assignments; 2) a tailored dataflow that exploits the ease of hybrid network to achieve faster DNN inference; and 3) an optimal pattern prediction method at very low cost to reduce DNN model deployment time. In conclusion, we have improved the performance of DNN accelerators and also achieved reductions in energy consumption. As compared to Simba, HPPI achieves average execution time reduction by 72.23% and average energy saving by 25.49%. As compared to ASCEND, HPPI achieves average time reduction by 34.04% and average energy saving by 10.34%.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[2] P. R. Bassi and R. Attux, "A deep convolutional neural network for COVID-19 detection using chest X-rays," *Res. Biomed. Eng.*, vol. 38, no. 1, pp. 139–148, 2022.

[3] S. Mozaffari, E. Arnold, M. Dianati, and S. Fallah, "Early lane change prediction for automated driving systems using multi-task attention-based convolutional neural networks," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 758–770, Sep. 2022.

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.

[5] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[6] R. Mayer and H.-A. Jacobsen, "Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools," *ACM Comput. Surveys*, vol. 53, no. 1, pp. 1–37, 2020.

[7] K.-C. Chen, M. Ebrahimi, T.-Y. Wang, and Y.-C. Yang, "NoC-based DNN accelerator: A future design paradigm," in *Proc. 13th IEEE/ACM Int. Symp. Netw.-Chip*, 2019, pp. 1–8.

[8] Y. S. Shao et al., "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2019, pp. 14–27.

[9] X. Liu, W. Wen, X. Qian, H. Li, and Y. Chen, "Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems," in *Proc. 23rd Asia–South Pacific Design Autom. Conf. (ASP-DAC)*, 2018, pp. 141–146.

[10] B. Zimmer et al., "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.

[11] M. Gao, X. Yang, J. Pu, M. Horowitz, and C. Kozyrakis, "TANGRAM: Optimized coarse-grained dataflow for scalable NN accelerators," in *Proc. 24th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, 2019, pp. 807–820.

[12] Z. Tan, H. Cai, R. Dong, and K. Ma, "NN-baton: DNN workload orchestration and chiplet granularity exploration for multichip accelerators," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2021, pp. 1013–1026.

[13] Y. Thonnart et al., "POPSTAR: A robust modular optical NoC architecture for chiplet-based 3D integrated systems," in *Proc. Design, Autom. Test Europe Conf. Exhibit. (DATE)*, 2020, pp. 1456–1461.

[14] 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 Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2332–2345, Oct. 2022.

[15] Y. Li, A. Louri, and A. Karanth, "SPACX: Silicon photonics-based scalable chiplet accelerator for DNN inference," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, 2022, pp. 831–845.

[16] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 7, pp. 2730–2741, Jul. 2022.

[17] Y. Li, A. Louri, and A. Karanth, "Scaling deep-learning inference with chiplet-based architecture and photonic interconnects," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 931–936.

[18] D. Vantrease et al., "Corona: System implications of emerging nanophotonic technology," *ACM SIGARCH Comput. Archit. News*, vol. 36, no. 3, pp. 153–164, 2008.

[19] S. Werner, J. Navaridas, and M. Luján, "Designing low-power, low-latency networks-on-chip by optimally combining electrical and optical links," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, 2017, pp. 265–276.

[20] E. Peter, A. Thomas, A. Dhawan, and S. R. Sarangi, "Active microring based tunable optical power splitters," *Opt. Commun.*, vol. 359, pp. 311–315, Jan. 2016.

[21] M. Wang, Y. Wang, C. Liu, and L. Zhang, "Network-on-interposer design for agile neural-network processor chip customization," in *Proc. 58th ACM/IEEE Design Autom. Conf. (DAC)*, 2021, pp. 49–54.

[22] A. Parashar et al., "Timeloop: A systematic approach to DNN accelerator evaluation," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2019, pp. 304–315.

[23] S. Li, D. Reddy, and B. Jacob, "A performance & power comparison of modern high-speed DRAM architectures," in *Proc. Int. Symp. Memory Syst.*, 2018, pp. 341–353.

[24] E. Taheri, S. Pasricha, and M. Nikdast, "ReSiPI: A reconfigurable silicon-photonic 2.5 D chiplet network with PCMs for energy-efficient interposer communication," in *Proc. 41st IEEE/ACM Int. Conf. Comput.-Aided Design*, 2022, pp. 1–9.

[25] A. Narayan, Y. Thonnart, P. Vivet, and A. K. Coskun, "PROWAVES: Proactive runtime wavelength selection for energy-efficient photonic NoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 10, pp. 2156–2169, Oct. 2021.

[26] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[28] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 7263–7271.
[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[30] Y. Ma, Y. Cao, S. Vrudhula, and J.-S. Seo, "Performance modeling for CNN inference accelerators on FPGA," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 4, pp. 843–856, Apr. 2020.
[31] E. Russo, M. Palesi, S. Monteleone, D. Patti, G. Ascia, and V. Catania, "LAMBDA: An open framework for deep neural network accelerators simulation," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Affiliated Events (PerCom Workshops)*, 2021, pp. 161–166.
[32] Q. Huang et al., "CoSA: Scheduling by constrained optimization for spatial accelerators," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2021, pp. 554–566.
[33] M. Kennedy and A. K. Kodi, "Laser pooling: Static and dynamic laser power allocation for on-chip optical interconnects," *J. Lightw. Technol.*, vol. 35, no. 15, pp. 3159–3167, Aug. 1, 2017.
[34] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2019, pp. 1–8.
[35] J. M. Wilson et al., "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 *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2018, pp. 276–278.
[36] H. Kwon, P. Chatarasi, M. Pellauer, A. Parashar, V. Sarkar, and T. Krishna, "Understanding reuse, performance, and hardware cost of DNN dataflow: A data-centric approach," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2019, pp. 754–768.
[37] Y. Thonnart et al., "A 10Gb/s Si-photonic transceiver with 150μW 120μs-lock-time digitally supervised analog microring wavelength stabilization for 1Tb/s/mm^2 die-to-die optical networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, 2018, pp. 350–352.

Guanglong Li received the B.S. degree in applied physics from Nanjing University, Nanjing, China, in 2021. He is currently pursuing the master's degree with the Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai, China.

His current research interests include photonic interconnect, network-on-chip, and DNN accelerator.

Yaoyao Ye (Member, IEEE) received the B.S. degree from the University of Science and Technology of China, Hefei, Anhui, China, in 2008, and the Ph.D. degree from Hong Kong University of Science and Technology, Hong Kong, in 2013.

She is currently an Associate Professor with the Department of Micro/Nano Electronics, Shanghai Jiao Tong University, Shanghai, China. She has authored/coauthored more than 50 papers in peer-reviewed journals and international conferences. Her research interests include multicore architecture, network-on-chip, and AI chip architecture.

Dr. Ye served for technical program committees of ASP-DAC 2024, DATE 2023, NOCS 2022–2023, NoCArc 2022–2023, ASP-DAC 2019–2021, ASP-DAC 2016–2017, and GLSVLSI 2017.