This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



# SPRINT: A High-Performance, Energy-Efficient, and Scalable Chiplet-based Accelerator with Photonic Interconnects for CNN Inference

Yuan Li, *Student Member, IEEE,* Ahmed Louri, *Fellow, IEEE,* Avinash Karanth, *Senior Member, IEEE*

**Abstract**—Chiplet-based convolution neural network (CNN) accelerators have emerged as a promising solution to provide substantial processing power and on-chip memory capacity for CNN inference. The performance of these accelerators is often limited by inter-chiplet metallic interconnects. Emerging technologies such as photonic interconnects can overcome the limitations of metallic interconnects due to several superior properties including high bandwidth density and distance-independent latency. However, implementing photonic interconnects in chiplet-based CNN accelerators is challenging and requires combined effort of network architectural optimization and CNN dataflow customization. In this paper, we propose SPRINT, a chiplet-based CNN accelerator that consists of a global buffer and several accelerator chiplets. SPRINT introduces two novel designs: (1) a photonic inter-chiplet network that can adapt to specific communication patterns in CNN inference through wavelength allocation and waveguide reconfiguration, and (2) a CNN dataflow that can leverage the broadcasting capability of photonic interconnects while minimizing the costly electrical-to-optical and optical-to-electrical signal conversions. Simulations using multiple CNN models show that SPRINT achieves up to 76% and 68% reduction in execution time and energy consumption, respectively, as compared to other state-of-the-art chiplet-based architectures with either metallic or photonic interconnects.

**Index Terms**—Convolution neural network, Chiplet, Accelerator, Photonic interconnects

## 1 Introduction

The ever increasing size of convolution neural network (CNN) models [1], [2], [3], [4] is driving the need to scale computing systems for higher processing power and on-chip memory capacity. As monolithic chip scaling slows down [5], [6], the chiplet-based architecture [7], [8] is considered a viable approach to continue the growth of computing system performance. Prior work [5] has explored performing inference of large-scale CNN models on chiplet-based accelerators. However, in such work, it has been shown that inter-chiplet metallic interconnects pose a major challenge to system performance due to excess latency and energy consumption [5]. This motivates us to explore other disruptive interconnect technologies for these chiplet-based accelerators.

Photonic interconnects can overcome the limitations of metallic interconnects due to superior properties such as high bandwidth density [9], [10] and distance-independent latency [11], [12], [13]. Photonic interconnects have been utilized in prior manycore architectures [14], [15], [16], [17], [18], [19]. However, implementing photonic interconnects in chiplet-based CNN accelerators requires combined effort of architectural optimization and dataflow customization. As stated above, prior photonic inter-chiplet networks [14], [15], [16], [17], [18], [19] target manycore architectures executing general applications, and consequently, often exhibit full connectivity and uniform bandwidth between chiplets to support diverse communication patterns observed in general applications. By contrast, the communication involved in CNN inference has several specific features such as non-uniform bandwidth demand between different chiplets, and recurrence of a few communication patterns [1], [2], [20], [21]. An optimized or a domain-specific photonic inter-chiplet network exploiting these specific features would significantly improve the performance, energy consumption, and scalability of chiplet-based CNN accelerators.

Additionally, the CNN dataflow should be customized to adapt to the unique properties of photonic interconnects. Most prior CNN dataflow optimizations [5], [22], [23], [24], [25], [26], [27], which are proposed for accelerators with only metallic interconnects, target reducing the data transmission distance by improving local data reuse. However, because of the distance-independent property of photonic interconnects, the data transmission distance across chiplets would not be a primary obstacle to system performance, possibly making the prior CNN dataflow optimizations less effective.

In this paper, we propose **SPRINT** - a chiplet-based accelerator with **S**ilicon **P**hotonic **R**econfigurable **INT**erconnects for CNN inference. The SPRINT architecture consists of (1) a photonic inter-chiplet network that connects a global buffer (GLB) and accelerator chiplets, and (2) an optically-enhanced and tailored CNN dataflow. Specifically, the photonic inter-chiplet network is optimized to adapt to the specific communication patterns in CNN inference through wavelength allocation and waveguide reconfiguration. The novel CNN dataflow exploits the inherent broadcast and multicast capabilities of photonic interconnects [28], [29], [30], while minimizing the costly electrical-to-optical (E/O) and optical-to-electrical (O/E) signal conversions [9], [31]. The

- *Yuan Li and Ahmed Louri are with the Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052. E-mail: {liyuan5859, louri}@gwu.edu.*
- *Avinash Karanth is with the School of Electrical Engineering and Computer Science, Ohio University, Athens, OH 45701. E-mail: karanth@ohio.edu.*

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



combined effects of the photonic inter-chiplet network and the tailored CNN dataflow result in significant reduction in execution time and energy consumption for CNN inference. We compare SPRINT with two state-of-the-art chiplet-based architectures using either an electrical mesh [5] or a photonic crossbar [18] for inter-chiplet communication. Simulation studies using multiple CNN models show that SPRINT achieves up to 76% and 68% reduction in execution time and energy consumption, respectively. Furthermore, when scaling the system to include 128 chiplets, the reduction in execution time and energy consumption increases to 78% and 83%, respectively, indicating the promising scalability of the proposed SPRINT architecture.

## 2 Background and Motivation

### 2.1 Communication in CNN Inference

CNN models often consist of a series of different layers (e.g., convolution layers, fully-connected layers, activation layers, etc.), where the convolution layers are most common and take a large fraction of the overall computations [32], [33], [34], [35]. In this paper, we focus on the processing of the convolution layers and fully-connected layers.

#### 2.1.1 Computation of a Convolution Layer

The computation of a convolution layer can be formulated as a multi-dimensional nested loop over weight kernels, input feature maps (ifmaps), and output feature maps (ofmaps). The dimensions include the height (R) and width (S) of the weight kernels, the height (E) and width (F) of the ofmaps, the number of input channels (C), and the number of weight kernels (M). The height (H) and width (W) of the ifmaps are not independent and can be derived from the previous dimensions. Fig. 1 (b) shows a nested loop example assuming each of the six dimensions (R, S, E, F, C, M) has a value of 2. Fig. 1 (a) presents the detailed computations in an iteration of the C loop. In this iteration, four weights (labeled 1, 2, 3, and 4) of each weight kernel and four input features (labeled a, b, d, and e) are transmitted to each computation unit. Within a computation unit, products of weights and corresponding input features are accumulated to a partial sum (psum) of an output feature. A complete output feature is obtained by adding up the psums of all iterations in the C loop. The above process is repeated to generate other output features as the weight kernels slide across the ifmaps.

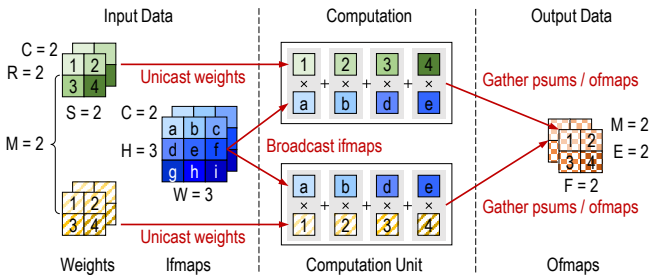
#### 2.1.2 Communication Patterns

Data communications incurred during the computation of a convolution layer include transmitting weights and input features to the computation units and gathering the psums or output features generated in the computation units back to memory. Unlike the diverse communication patterns in general applications, the communication patterns in CNN accelerators are fairly regular and largely determined by the dimension values of the nested loop, the parameters of the computing system (e.g., the number of computation units), and the dataflow utilized. Specifically, the communications in CNN accelerators exhibit the following three features:

**Non-uniform Bandwidth Demand**: Assume that the input data (weights and input features) are initially located in the GLB and the generated output data (psums or output features) are transmitted back to the GLB. The bandwidth needed for data exchange between the GLB and the computation units is often non-uniform. As shown in Fig. 1 (a), in one iteration of the C loop, 13 data elements (4 weights and 9 input features) are transmitted from the GLB to a computation unit while only 4 data elements (4 psums) are transmitted from a computation unit to the GLB. Meanwhile, no data is exchanged between the two computation units. As such, prior inter-chiplet networks [15], [16], [18] which provide equal per-chiplet bandwidth would clearly be inefficient, if applied to CNN accelerators with GLB and computation units on different chiplets.

**Recurrence of Communication Patterns**: As shown in Fig. 1 (a), there are three communication patterns in the C loop: unicast of weights, broadcast of input features, and gathering of psums or output features. These communication patterns recur in each iteration of the outer E and F loops. A natural approach to efficiently support the recurrent communication patterns is to design a network that can dynamically switch between multiple configurations, each of which adapts to a specific communication pattern.

**Prevalent Broadcasting**: Broadcast communication is prevalent in the computation of convolution layers [2], as a large fraction of computations often share the same input data. As shown in Fig. 1 (a), the input features are broadcast to two separate computation units, because each computation unit processes one weight kernel and both weight kernels slide across the same ifmaps. Although prior dataflow optimizations also exploit broadcast and multicast [5], [22], [23], [24], [25], [26], [27], these operations are often very costly to implement in metallic interconnects in terms of latency, energy consumption, and overall circuity area.




(a) Computations in one iteration in C loop

```
1 for E = [ 0, 2 )
2   for F = [ 0, 2 )
3     for C = [ 0, 2 )
4       for M = [ 0, 2 )
5         for R = [ 0, 2 )
6           for S = [ 0, 2 )
7             Ofmaps [M, E, F] += Weights [M, C, R, S] * Ifmaps [C, R + E – 1, S + F - 1]
```

(b) Nested loop representation

Fig. 1: Computations in a sample convolution layer and the corresponding nested loop representation. Communications incurred include unicasting weights, broadcasting ifmaps, and gathering psums or ofmaps.

### 2.2 Photonic Interconnects

In this subsection, we introduce basic optical components and discuss the drawbacks of existing photonic inter-chiplet networks when applied to chiplet-based CNN accelerators.

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



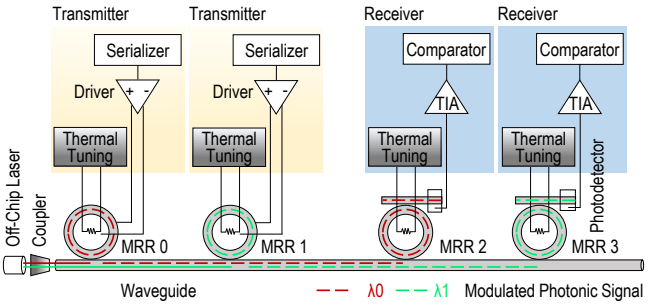



Fig. 2: A wavelength division multiplexing photonic link.

### 2.2.1 Photonic Interconnects

Fig. 2 presents a photonic link with wavelength division multiplexing (WDM). In this example, an off-chiplet laser emits two optical signals with different wavelengths, $\lambda 0$ and $\lambda 1$. The optical signals are then coupled into a waveguide using an optical coupler [9]. At the transmission side, two micro-ring resonators (MRRs) labeled MMR0 and MMR1 are used as optical modulators to separately modulate input signals at wavelengths $\lambda 0$ and $\lambda 1$. At the receiving side, another two MRRs labeled MMR2 and MMR3 are used as optical filters to select a specific modulated wavelength. The selected wavelength is detected by a photodetector [9] and converted back to an electrical signal. The electrical signals are then amplified by the transimpedance amplifiers (TIAs) and forwarded to comparators to retrieve the original data transmitted. A MRR, used as either an optical modulator or filter, is tuned by a resistive heater controlled by a thermal tuning unit to mitigate thermal and process variations [9], [12]. Two or more wavelengths can be multiplexed onto the same waveguide using WDM technique.

### 2.2.2 Photonic Inter-chiplet Networks

Several photonic inter-chiplet networks [14], [15], [16], [17], [18], [19] have been proposed recently. However, these networks are implemented in systems with CPU/GPU chiplets running general manycore applications. The resulting uniform bandwidth resource allocation and full connectivity between chiplets lead to excessive implementation costs. For example, [16], [18], [19] propose a photonic crossbar using single-write multiple-read (SWMR) channels. The number of required MRRs in a photonic crossbar scales quadratically with the number of chiplets, leading to excessive area cost and energy consumption [9], [28], [29]. By contrast, the proposed SPRINT architecture is designed specifically for CNN inference application, which results in an optimized photonic inter-chiplet network with much fewer MRRs.

## 3 SPRINT Architecture

SPRINT architecture consists of one GLB and several accelerator chiplets integrated in a package. A reconfigurable photonic inter-chiplet network is designed to provide communication between the GLB and other accelerator chiplets. The photonic inter-chiplet network supports (1) data transmission from the GLB to accelerator chiplets, and (2) data gathering from accelerator chiplets to the GLB.

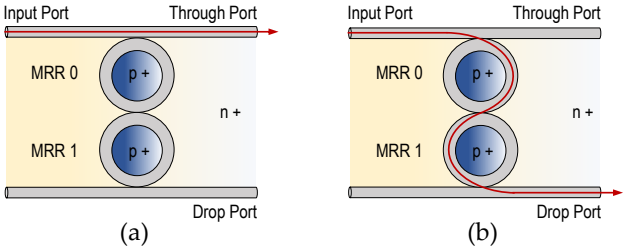



Fig. 3: The electrical-optical switch in (a) off or (b) on state.

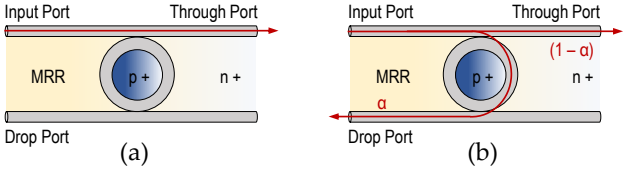



Fig. 4: The tunable splitter in (a) disabled state or (b) with a split ratio of $\alpha/(1-\alpha)$.

### 3.1 Photonic Inter-chiplet Network

#### 3.1.1 Components for Network Reconfiguration

We introduce two additional optical components utilized in the SPRINT photonic inter-chiplet network to support different communication patterns from the GLB to accelerator chiplets: electrical-optical switch [36] and tunable splitter [37].

**Electrical-optical Switch**: SPRINT includes 1×2 electrical-optical switches, each of which is associated to an accelerator chiplet. As shown in Fig. 3, a 1×2 electrical-optical switch consists of two waveguides (one with the input and through ports while the other one with the drop port) and two MRRs (labeled MRR0 and MRR1). The two MRRs are positioned between the two waveguides. Regions inside and outside the MRRs are p-type and n-type semiconductor regions, respectively, to form the PIN diode structure. Switching an optical signal between the through port and the drop port is accomplished with the switching of MRR resonance using the free-carrier dispersion effect [36]. When the MRRs are at off-resonance as shown in Fig. 3 (a), the optical signal from the input port is directly forwarded to the through port. When the MRRs are at on-resonance as shown in Fig. 3 (b), the optical signal from the input port is guided through two MRRs to the drop port. In SPRINT, these 1×2 electrical-optical switches are utilized to either combine or separate multiple waveguides.

**Tunable Splitter**: Another component included in SPRINT is the tunable splitter [37]. Different from MRRs working at on-resonance and off-resonance states as optical modulators or filters, a tunable splitter works in the transient zone between on-resonance and off-resonance. As shown in Fig. 4, the regions inside and outside the MRR are doped to form the PIN diode structure. When applying an appropriate voltage to the PIN diode structure, the optical signal from the input port is split into two parts and guided to the drop port ($\alpha$ fraction) and through port ($(1-\alpha)$ fraction), respectively. By tuning the applied voltage (0 to 5 volts), split ratios in the range of 0.4 to 1.8 can be obtained [37]. Digital-to-analog converters (DACs) are utilized to accurately adjust

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



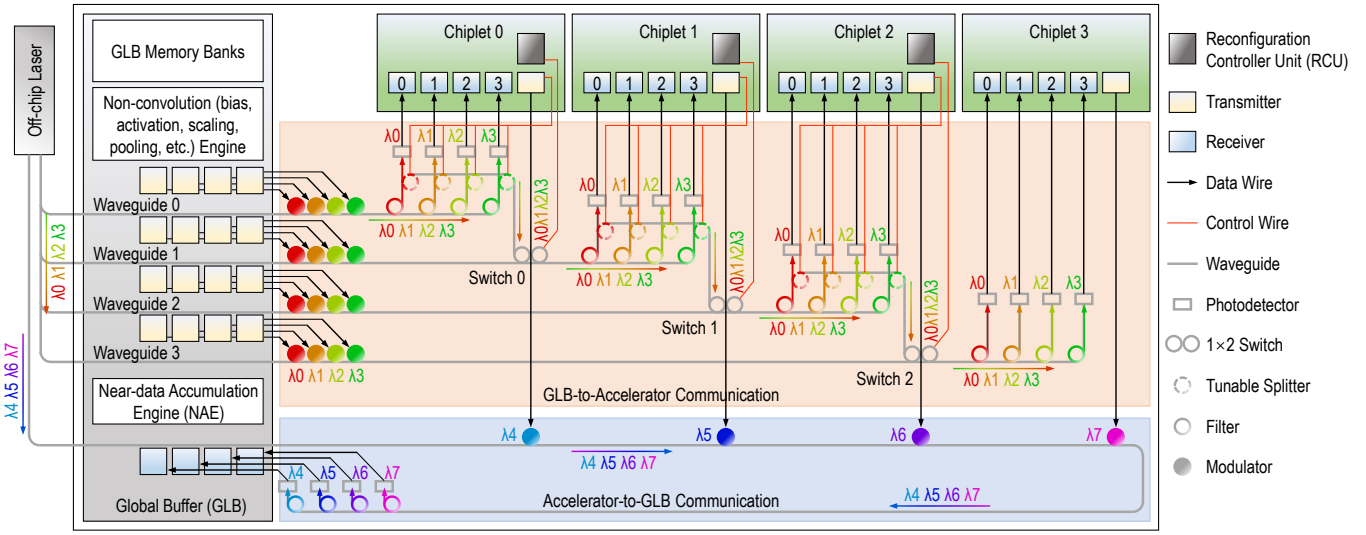



Fig. 5: A small-scale SPRINT architecture. We assume four accelerator chiplets, each with four receivers and one transmitter. Eight wavelengths are multiplexed in a photonic inter-chiplet network. Wavelengths $\lambda 0$, $\lambda 1$, $\lambda 2$, $\lambda 3$ are responsible for data transmission from the GLB to all other accelerator chiplets, while wavelengths $\lambda 4$, $\lambda 5$, $\lambda 6$, $\lambda 7$ are responsible for data transmission from a corresponding accelerator chiplet to the GLB. The legends illustrate some key components and wires. An optical component with a specific color means this component is associated with one wavelength only.

the applied voltage. In SPRINT, a tunable splitter is utilized to split an appropriate portion of the power of an optical signal for detection purpose while forwarding the rest of the optical signal to downstream locations. We cascade multiple tunable splitters [38] in cases where split ratios at a wider range are necessary.

### 3.1.2 Photonic Inter-chiplet Network

For clarity, we illustrate the photonic inter-chiplet network of a scale-down SPRINT architecture that consists of a GLB and four accelerator chiplets in Fig. 5. The photonic inter-chiplet network consists of two parts: (1) GLB-to-accelerator communication and (2) accelerator-to-GLB communication. In this example, eight wavelengths are utilized for the inter-chiplet communication. All the optical components (e.g., waveguides, MRRs, photodetectors, electrical-optical switches, etc.) are implemented on a silicon interposer [39] using CMOS compatible process, while all the electrical peripheral circuits are implemented on the silicon dies of the GLB and accelerator chiplets. In this subsection, we describe the architectural innovations in SPRINT photonic inter-chiplet network including wavelength allocation for non-uniform bandwidth demand, and waveguide reconfiguration for recurrent communication patterns. We further discuss the physical implementation of this network.

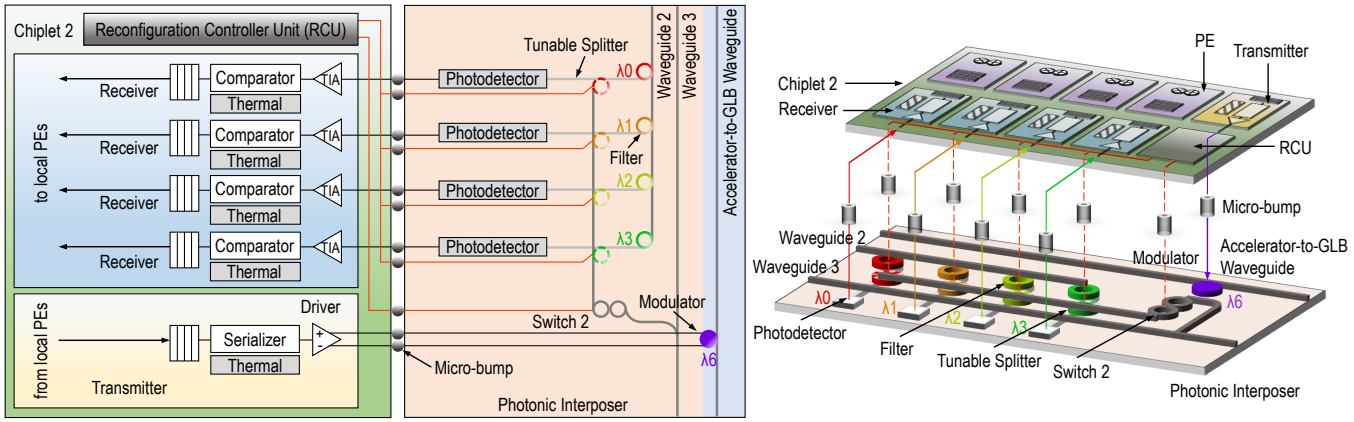
**Wavelength Allocation**: We divide all the available wavelengths into two groups - one group for data transmission from the GLB to accelerator chiplets ($\lambda 0$, $\lambda 1$, $\lambda 2$, $\lambda 3$ in Fig. 5), while the other group for data gathering from accelerator chiplets to the GLB ($\lambda 4$, $\lambda 5$, $\lambda 6$, $\lambda 7$ in Fig. 5). Although wavelengths are equally divided into the two groups in the example shown in Fig. 5, this is not the case in the full-scale SPRINT architecture, where wavelengths are non-uniformly allocated based on the specific bandwidth demand. In Section 4, we demonstrate that the bandwidth demand may vary when different CNN dataflows and chiplet-level architectures are implemented, or when different convolution layers are processed, as these factors significantly impact the transmission of weights, input features, and psums. Please note that the number of wavelengths in the group for data gathering from accelerator chiplets to the GLB is proportional to the number of accelerator chiplets in the system, as each accelerator chiplet is assigned a unique wavelength (e.g., wavelength $\lambda 4$ is assigned to `Chiplet 0` in Fig. 5).

**GLB-to-Accelerator Communication**: This communication, represented by the upper part of Fig. 5, is used to transmit weights and input features from the GLB to accelerator chiplets. Due to the recurrent unicast and broadcast communication patterns shown in Fig. 1 and broadly observed in CNN inference [2], we propose three communication modes for the data transmission from the GLB to accelerator chiplets, namely unicast mode, broadcast mode, and hybrid mode. The unicast mode is used to simultaneously transmit exclusive data from the GLB to each individual accelerator chiplet with equal bandwidth. As shown in Fig. 5, four separate waveguides connect the GLB with corresponding accelerator chiplets (e.g., `Waveguide 0` connects the GLB with `Chiplet 0`), forming four dedicated communication channels. To eliminate interference, the 1×2 electrical-optical switches connecting adjacent waveguides (e.g., `Switch 0` that connects `Waveguide 0` and `Waveguide 1`) are tuned at off-resonance. Further, the tunable splitters are all disabled since there are no accelerator chiplets sharing the same communication channel. Data is transmitted from the GLB to an accelerator chiplet using wavelengths $\lambda 0$, $\lambda 1$, $\lambda 2$, and $\lambda 3$.

The broadcast mode is used to transmit the shared data from the GLB to all accelerator chiplets in the system. As shown in Fig. 5, all four waveguides (`Waveguide 0-3`) are combined into a SWMR channel by tuning 1×2 electrical-optical switches (`Switch 0-2`) at on-resonance. The tunable splitters are enabled and tuned to different split ratios based

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015






(a) Schematic view (b) Physical layout

Fig. 6: The physical implementation of the SPRINT photonic inter-chiplet network, taking `Chiplet 2` in Fig. 5 as an example. The legends utilized here are compatible with the ones in Fig. 5.

on their positions in the SWMR channel. As the example shown in Fig. 5, tunable splitters attached to `Chiplet 0`, `Chiplet 1`, and `Chiplet 2` are tuned to split ratios of `3/1`, `2/1`, and `1/1`, respectively, based on the number of downstream accelerator chiplets along the channel. Please note that there are no tunable splitters attached to the last accelerator chiplet (`Chiplet 3`). Data is broadcast from the GLB to all accelerator chiplets using wavelengths $\lambda 0$, $\lambda 1$, $\lambda 2$, and $\lambda 3$.

The hybrid mode works as a combination of previous two communication modes. In this mode, the waveguides (`Waveguide 0-3`) are grouped into multiple segmented SWMR or dedicated communication channels. Some $1\times2$ electrical-optical switches are tuned at off-resonance while the others are tuned at on-resonance to separate segmented channels and maintain intra-segment connectivity, respectively. The tunable splitters are tuned to different split ratios based on their positions in the corresponding segmented channel. In the example shown in Fig. 5, we can create two segmented SWMR channels, each connecting the GLB to a set of two accelerator chiplets (`Chiplet 0/1` and `Chiplet 2/3`), by tuning `Switch 0` and `Switch 2` 2 at on-resonance and `Switch 1` at off-resonance. Accordingly, tunable splitters attached to `Chiplet 0` and `Chiplet 2` are tuned to split ratio of 1/1, while tunable splitters attached to `Chiplet 1` are disabled. Further, by tuning `Switch 2` at off-resonance and disabling tunable splitters attached to `Chiplet 2`, a previous segmented SWMR channel is divided into two dedicated communication channels. The hybrid mode is introduced to support more sophisticated communication patterns observed in CNN inference other than unicast and broadcast. It is particularly useful in cases when accelerator chiplets in the system are not fully occupied, or when multiple CNN layers with distinct communication patterns are processed in a pipelined manner.

The above three communication modes for GLB-to-accelerator communication can be tuned at runtime by setting appropriate switching signals and split ratios on $1\times2$ electrical-optical switches and tunable splitters, respectively. In Section 4.2, we will present how unicast and broadcast modes are utilized in turn for transmission of weights and input features, respectively, when different chiplet-level accelerator architectures and dataflows are assumed.

**Accelerator-to-GLB Communication**: This communication, represented by the lower part of Fig. 5, is used to collect the psums and output features from the accelerator chiplets and transmit them to the GLB. The photonic inter-chiplet network works as a multiple-write single-read (MWSR) channel. Each accelerator chiplet is assigned a specific wavelength (e.g., $\lambda 4$ is assigned to `Chiplet 0` in Fig. 5). Psums and output features generated by different accelerator chiplets are transmitted on different wavelengths ($\lambda 4$, $\lambda 5$, $\lambda 6$, $\lambda 7$ in Fig. 5) and obtained by receivers at the GLB side. The received psum are accumulated in the near-data accumulation engine (NAE) and stored in GLB for future reference, while output features go through bias and activation functions and act as the input data of the next CNN layer.

### 3.1.3 *Physical Implementation*

Fig. 6 depicts the physical implementation of SPRINT photonic inter-chiplet network corresponding to the SPRINT architecture shown in Fig. 5. For simplicity, we only present `Chiplet 2` and its associated electrical and optical components. The physical implementation of the entire SPRINT photonic inter-chiplet network can be easily inferred.

As shown in Fig. 6, the electrical circuits of transmitters and receivers, the reconfiguration controller unit (RCU), and multiple processing elements (PEs) of `Chiplet 2` are integrated on a separate silicon die and connected to the photonic interposer through micro-bumps [39]. RCU is used to setup the communication mode for GLB-to-accelerator communication from three available modes discussed in Section 3.1.2. RCU switches the $1\times2$ electrical-optical switch between on-resonance and off-resonance, and tunes associated tunable splitters to appropriate split ratios with DACs. The micro-bumps are used for (1) RCU and thermal tuning signal transmission from an accelerator chiplet to the photonic interposer, and (2) data exchange between the photonic interposer and accelerator chiplets. We only show the RCU tuning signal transmission (dashed lines) and data exchange between the photonic interposer and the accelerator chiplet (solid lines) in Fig. 6 (b), for simplicity's sake.

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



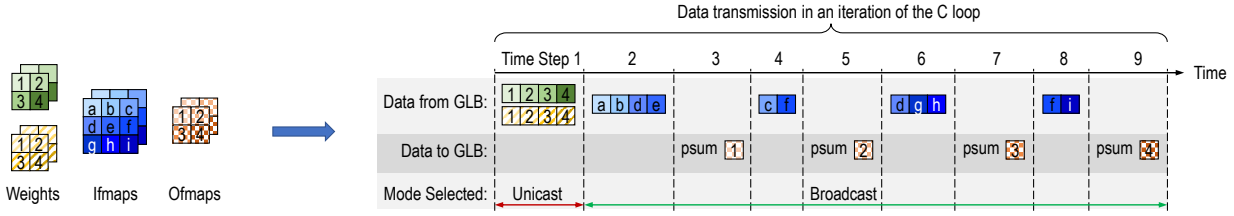



Fig. 7: SPRINT GLB-to-accelerator communication mode selection when using weight-stationary [5] dataflow. We present data transmission in one iteration of the `C` loop and the selected communication modes.

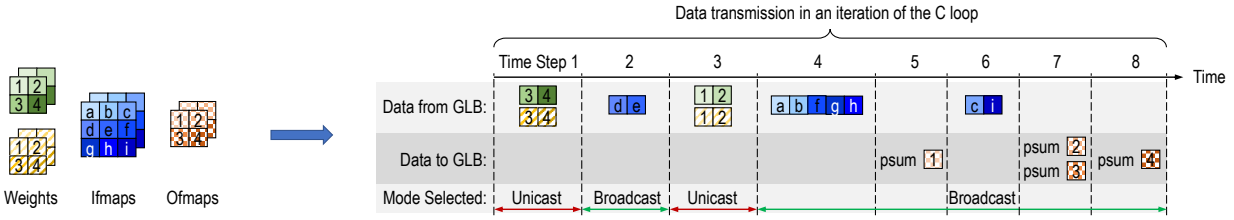



Fig. 8: SPRINT GLB-to-accelerator communication mode selection when using row-stationary [22] dataflow. We present data transmission in one iteration of the `C` loop and the selected communication modes.

All optical components (e.g., waveguide, modulator, filter, photodetector, tunable splitter, and 1×2 electrical-optical switch) are integrated on the photonic interposer. There are mainly three waveguides shown in Fig. 6: `Waveguide 2` and `Waveguide 3` that connect the GLB to `Chiplet 2` and `Chiplet 3`, respectively, and another waveguide for accelerator-to-GLB communication. `Waveguide 2` are connected to `Waveguide 3` through the drop port of `Switch 2`. These main waveguides are located in the same layer in the interposer. Modulators, filters, tunable splitters, and 1×2 electrical-optical switches are vertically coupled [40] to the above waveguides. The drop port of a tunable splitter is connected to the corresponding photodetector through another waveguide that is located in a separate layer [41] in the interposer from previously discussed main waveguides, to avoid waveguide crossing.

We present the working flow of the components involved in Fig. 6 (b), by taking the broadcast mode in GLB-to-accelerator communication as an example. Modulated wavelengths $\lambda 0$, $\lambda 1$, $\lambda 2$, and $\lambda 3$ in `Waveguide 2` are selected by vertically coupled filters and guided to separate tunable splitters. Since the broadcast mode is enabled and there is only one downstream accelerator chiplet after `Chiplet 2`, the split ratio of the tunable splitters are tuned to 1/1. The power of each wavelength is split into two equal portions: one portion from the drop port of the tunable splitter is guided to a corresponding photodetector for detection while the other portion from the through port is collected and merged into `Waveguide 3` through `Switch 2`. The photocurrent signals from photodetectors are transmitted through micro-bumps to receivers and converted to transmitted data.

### 3.2 Chiplets in SPRINT Architecture

**GLB**: Two additional modules, NAE and non-convolution engine, are integrated on the GLB silicon die. NAE performs accumulation operations on psums collected from accelerator chiplets. In the prior chiplet-based CNN accelerator [5], an accelerator chiplet may forward the psums locally generated to a remote accelerator chiplet for cross-chiplet accumulation, possibly leading to communication between two arbitrary accelerator chiplets. In this case, costly E/O and O/E signal conversions are inevitable if a photonic inter-chiplet network is used. By exploiting the distance-independent property of photonics, psums generated in accelerator chiplets are collected and transmitted back to NAE for accumulation, significantly reducing the E/O and O/E signal conversions. The non-convolution engine is used to process the non-convolution layers in CNN models by performing functions such as bias, activation, scaling, and pooling.

**Accelerator Chiplet**: An accelerator chiplet consists of an RCU, electrical circuits for transmitters and receivers, and multiple PEs as shown in Fig. 5. RCU is responsible for tuning the 1×2 electrical-optical switch and tunable splitters, as we have discussed in Section 3.1.3. The respective numbers of circuits for transmitters and receivers are determined by the number of wavelengths used for GLB-to-accelerator and accelerator-to-GLB communications. In Section 4.2, We will show that the respective numbers of wavelengths for GLB-to-accelerator and accelerator-to-GLB communications may vary when different chiplet-level architectures or underlying dataflows are assumed.

## 4 SPRINT Dataflow Customization

### 4.1 Package-level Data Partition

The conventional CNN dataflow optimizations [5], [22], [23], [24], [25], [26], [27] are proposed for accelerators with only metallic interconnects. Such dataflow optimizations often target reducing data transmission distance by improving local data reuse. As SPRINT architecture relies on a photonic inter-chiplet network with distance-independent latency, prior dataflow optimizations would not necessarily be effective. This is because (1) data transmission distance would not be a primary obstacle to system performance, and

(2) new unaddressed factors (e.g., utilization of broadcast and multicast communications, the number of E/O and O/E signal conversions, etc.) may have significant impact on system performance.

At the package level, we spatially distribute the computations of each weight kernel in the M dimension to an accelerator chiplet and temporally iterate the computations of each input channel in the C dimension. In doing so, the same input feature would be consumed by all involved accelerator chiplets, leading to more opportunities for broadcasting or multicasting communication. Meanwhile, the accumulation computations along the C dimension are confined in each individual accelerator chiplet, reducing cross-chiplet accumulations, hence, reducing the number of required E/O and O/E signal conversions.

### 4.2 Chiplet-level Dataflow

In this subsection, we discuss different configurations of the SPRINT photonic inter-chiplet network when assuming different chiplet-level accelerator architectures with weight-stationary (WS) [5] or row-stationary (RS) [22] CNN dataflow. In Section 6, we present the simulation results when assuming two additional CNN dataflows, namely output-stationary (OS) dataflow [23] and no-local-reuse (NLR) dataflow [27].

#### 4.2.1 Accelerator Chiplet with Weight-stationary Dataflow

We assume that the accelerator chiplet architecture is similar to [5] and WS CNN dataflow is adopted. The circuits for transmitters and receivers and the local PEs are connected by a 2D electrical mesh intra-chiplet network as in [5]. Fig. 7 presents the data transmission in an iteration of the C loop, using the same input CNN layer as in Fig. 1. The working mode of GLB-to-accelerator communication switches regularly based on the type of the transmitted data.

In Time Step 1, weights from the same input channel (C dimension) but different weight kernels (M dimension) are transmitted from the GLB to different accelerator chiplets using the unicast mode. Since WS dataflow keeps weights stationary within a PE, no other weights need to be transmitted before the completion of current iteration of the C loop. Starting from Time Step 2, the GLB broadcasts input features to two involved accelerator chiplets. In Time Step 4, 6, and 8, different input features are broadcast (assume that the local ifmap buffer in a PE can hold all the received input features for local reuse). Consequently, the broadcast mode for GLB-to-accelerator communication is selected and maintained. In Time Step 3, 5, 7, and 9, psums of corresponding output features are either stored for future intra-chiplet accumulation or transmitted back to the GLB using accelerator-to-GLB communication. Time Step 9 indicates the completion of the current iteration of the C loop. Similar computations and data transmission are performed for the next input channel. The final output features are obtained by accumulating the corresponding psums in all iterations of the C loop. For each iteration of the C loop, GLB-to-accelerator communication is switched between unicast and broadcast modes once. Please note that it is not necessary to switch communication mode this frequently as in Fig. 7, as long as there is sufficient buffer capacity to hold required input data and intermediate results.

The number of wavelengths for GLB-to-accelerator communication and the number of wavelengths for accelerator-to-GLB communication are determined by the accelerator chiplet architecture and the dataflow utilized. In the case shown in Fig. 7, each local PE requires specific weights and input features for computation while the generated psums can be accumulated across local PEs, leading to a peak bandwidth demand ratio of 4:1 for GLB-to-accelerator and accelerator-to-GLB communications. Consequently, we allocate 80% of the total wavelengths for GLB-to-accelerator communication. This allocation may vary when the chiplet-level architecture or dataflow is altered.

#### 4.2.2 Accelerator Chiplet with Row Stationary Dataflow

We assume that the accelerator chiplet architecture is similar to [22] and RS CNN dataflow is adopted. Multiple X-buses and the circuits for transmitters and receivers are connected by a Y-bus, while each X-bus is used to connect a group of local PEs as in [22]. Fig. 8 presents the data transmission in an iteration of the C loop, using the same input CNN layer as in Fig. 1. The working mode of GLB-to-accelerator communication switches regularly, but in a different pattern from the one described in Section 4.2.1, indicating the significant impact of chiplet-level accelerator chiplet architecture and underlying dataflow on inter-chiplet communication.

In Time Step 1, two rows of weights from the same input channel (C dimension) but different weight kernels (M dimension) are transmitted from the GLB to different accelerator chiplets using the unicast mode. In Time Step 2, a portion of the second row of the input features (labeled d and e) is broadcast to two involved accelerator chiplets using the broadcast mode. In Time Step 3, GLB-to-accelerator communication is switched back to the unicast mode to transmit another two rows of weights from the same input channel but different weight kernels. In Time Step 4 and 6, the broadcast mode is selected and maintained to respectively transmit input features labeled a, b, f, g, h and input features labeled c and i. The transmission of input features follow the exact pattern as in [22], which enables input feature reuse between PEs along diagonals. In Time Step 5, 7, and 8, psums of corresponding output features are generated and transmitted back to the GLB using accelerator-to-GLB communication. Time Step 8 indicates the completion of the current iteration of the C loop. Similar computations and data transmission are performed for other iterations of C loop until final output features are obtained.

According to [22], the weights and input features are reused along horizontal and diagonal directions while the psums are accumulated along the vertical direction, making a peak bandwidth demand ratio of 3:1 for GLB-to-accelerator and accelerator-to-GLB communications. Consequently, we allocate 75% of the total wavelengths for GLB-to-accelerator communication. The allocation is different from Section 4.2.1 because a different chiplet-level dataflow is utilized.

## 5 Evaluation Methodology

We compare the SPRINT architecture, in terms of execution time and energy consumption, with other two state-of-the-art chiplet-based architectures with either an

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



TABLE 1: SPRINT Architecture Parameters

| | | |
|---|---|---|
| **Package** | Number of chiplets | 64 |
| | Global buffer | 128 KiB / chiplet |
| | Inter-chiplet bandwidth | 800 Gbps / chiplet |
| | Data rate per wavelength | 10 Gbps |
| **WS Chiplet** | Number of PEs | 16 |
| | Chiplet-level network | 2D mesh |
| | Number of vector MACs | 8 |
| | Vector MAC width | 8 |
| | Weight buffer | 64 KiB / PE |
| | Ifmap buffer | 6 KiB / PE |
| | Accumulation buffer | 2 KiB / PE |
| **RS Chiplet** | Number of PEs | 168 |
| | Chiplet-level network | X-bus / Y-bus |
| | Weight buffer | 448 B / PE |
| | Ifmap buffer | 24 B / PE |
| | Accumulation buffer | 48 B / PE |
| **OS Chiplet** | Number of PEs | 64 |
| | Chiplet-level network | dedicated links |
| | Nbin (input feature buffer) | 1 KiB |
| | Nbout (output feature buffer) | 1 KiB |
| | SB (synapse buffer) | 2 KiB |
| **NLR Chiplet** | Number of PEs | 64 |
| | Chiplet-level network | dedicated links |
| | Nbin (input feature buffer) | 16 B |
| | Nbout (output feature buffer) | 16 B |
| | SB (synapse buffer) | 256 B |

TABLE 2: Photonic Parameters

| Component | Value | Component | Value |
|---|---|---|---|
| Laser source | 5 dB [42] | Ring drop | 1 dB [43] |
| Coupler | 1 dB [42] | Ring through | 0.01 dB [43] |
| Waveguide | 1 dB/cm [42] | Photodetector | 0.1 dB [42] |
| Splitter | 0.2 dB [44] | Waveguide-to-receiver | 0.5 dB [45] |
| Waveguide bend | 1 dB [45] | Receiver sensitivity | -26 dBm [42] |
| Waveguide crossover | 0.05 dB [45] | Ring heating | 0.32 mW [46] |

electrical mesh [5] or a photonic crossbar [18] for inter-chiplet communication. Meanwhile, we explore four different chiplet-level architectures and dataflows and their impact on system performance. Table 1 lists the key architectural parameters of the SPRINT architecture and four different chiplet-level architectures that we have explored. Different chiplet-level architectures may lead to distinct bandwidth demands for different types of data involved, as explained in Section 4.2. For each chiplet-level architecture selected, a specific wavelength allocation process discussed in Section 3.1.2 is performed and the resulting SPRINT architecture is utilized for evaluation.

**Simulators**: In order to simulate chiplet-based architectures, We extend the open-source Timeloop simulator to support the non-uniform distribution of latency and bandwidth between PEs. The execution time is derived from the computation time and the communication time, taking the overlap between computation and communication into account. The extended simulator tracks the number of arithmetic operations and the number of accesses to each on-package memory hierarchy to calculate the computation time and on-package communication time, respectively. The calculation takes the hierarchical network architecture (inter-chiplet and intra-chiplet networks) into account and ensures that data transmission does not exceed the bandwidth limit of the corresponding link. The delay for tuning the 1×2 electrical-optical switches and tunable splitters is set to 500 $ps$ [37]. The off-package communication time (access time to off-package DRAM) is obtained from the DRAMSim2 simulator [47].

**Power Model**: We evaluate the power consumption of computations both on the accelerator chiplets and in NAE using `Synopsys Design Compiler`. The power consumption of accessing on-package memory hierarchies and off-package DRAM is obtained using CACTI 6.0 [48] and DRAMSim2, respectively. The power consumption of on-package metallic interconnects are obtained using DSENT [43], while the power consumption of photonic interconnects in SPRINT and the photonic crossbar [18] is derived from Equation (1):

$$P_{total} = P_{laser} + P_{TX} + P_{RX} \quad (1)$$

The overall power consumption $P_{total}$ consists of three parts: laser power $P_{laser}$, power consumption of transmitting circuitry $P_{TX}$, and power consumption of receiving circuitry $P_{RX}$. We calculate $P_{TX}$ and $P_{RX}$ using the same parameters as in [49] and scale the results to 28 $nm$ technology [49], [50]. Please note that the power consumption for ring heating has been included in both $P_{TX}$ and $P_{RX}$. The values for $P_{TX}$ and $P_{RX}$ are 1.22 $mW$ and 0.92 $mW$, respectively when a moderate 0.32 $mW$ [46] ring-heating power consumption is assumed. Laser power $P_{laser}$ can be expressed by three terms: photodetector sensitivity $P_{rs}$, insertion loss $C_{loss}$, and system margin $M_{system}$ as shown in Equation (2):

$$P_{laser} = P_{rs} + C_{loss} + M_{system} \quad (2)$$

We obtain the photodetector sensitivity $P_{rs}$ and insertion loss $C_{loss}$ from parameters listed in Table 2. The system margin $M_{system}$ is assumed to be 4 $dB$ [28]. From Equation (1 and 2) and parameters from [49] and Table 2, we obtain the energy consumption of the SPRINT photonic inter-chiplet network to be 0.77 $pJ/bit$, indicating the superior energy consumption of photonic interconnects.

**Architectures for Comparison**: SPRINT architecture is compared with two state-of-the-art chiplet-based architectures with electrical mesh [5] or photonic crossbar [18] for inter-chiplet communication. These three inter-chiplet networks are evaluated when four different chiplet-level accelerator chiplet architectures and dataflows shown in Table 1 are assumed. Please note that the photonic crossbar in [18] is originally designed to connect CPU/GPU chiplets. We replace the CPU/GPU chiplets with accelerator chiplets for fair comparison. We largely use similar chiplet-level configurations as in the original papers for WS chiplet [5], RS chiplet [22], OS chiplet [23], and NLR chiplet [27]. The

TABLE 3: Convolution and Fully-connected Layers

| Label | Layer Name | Comp. / Comm. | Label | Layer Name | Comp. / Comm. |
|---|---|---|---|---|---|
| | VGG-16 [35] | | L4 | res2[a-c]_branch2c | 469.7 |
| L1 | conv1-1 | 1117.4 | L5 | res2[b-c]_branch2a | 125.2 |
| L2 | conv1-2 | 1137.9 | L6 | res3a_branch1 | 878.6 |
| L3 | conv2-1 | 2108.1 | L7 | res3a_branch2a | 245.5 |
| L4 | conv2-2 | 2109.1 | L8 | res3[a-d]_branch2b | 932.6 |
| L5 | conv3-1 | 2654.2 | L9 | res3[a-d]_branch2c | 617.0 |
| L6 | conv3-2 | 2655.3 | L10 | res3[b-d]_branch2a | 219.9 |
| L7 | conv4-1 | 1339.2 | L11 | res4a_branch1 | 887.2 |
| L8 | conv4-2 | 1339.6 | L12 | res4a_branch2a | 385.6 |
| L9 | conv5-1 | 375.9 | L13 | res4[a-f]_branch2b | 361.2 |
| L10 | fc-4096 | 2.0 | L14 | res4[a-f]_branch2c | 328.4 |
| L11 | fc-4096 | 2.0 | L15 | res4[b-f]_branch2a | 221.9 |
| L12 | fc-4096 | 2.0 | L16 | res5a_branch1 | 357.6 |
| | | | L17 | res5a_branch2a | 283.3 |
| | ResNet-50 [32] | | L18 | res5[a-c]_branch2b | 97.0 |
| L1 | conv1 | 5882.9 | L19 | res5[a-c]_branch2c | 95.6 |
| L2 | res2a_branch2a | 124.5 | L20 | res5[b-c]_branch2a | 89.4 |
| L3 | res2[a-c]_branch2b | 972.4 | L21 | fc-1000 | 2.0 |

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



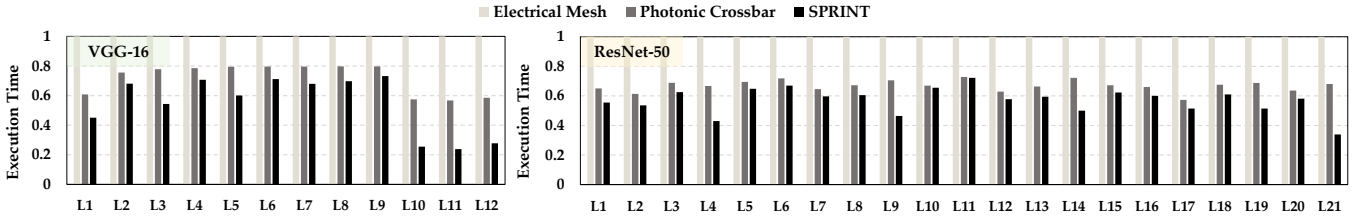



Fig. 9: Execution time comparison across different VGG-16 and ResNet-50 layers when utilizing WS accelerator chiplets. In each layer, all values are normalized to the electrical mesh architecture.

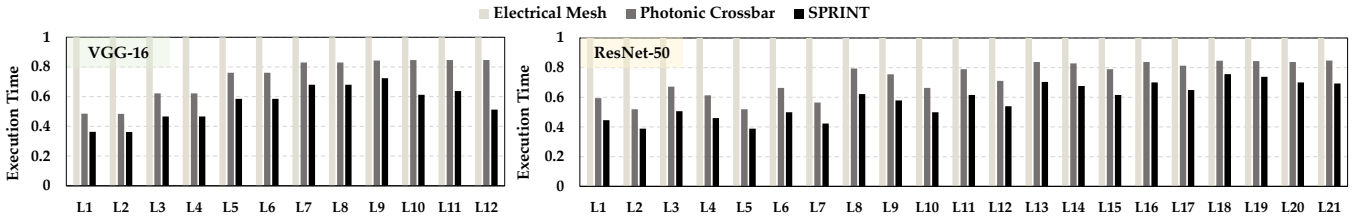



Fig. 10: Execution time comparison across different VGG-16 and ResNet-50 layers when utilizing RS accelerator chiplets. In each layer, all values are normalized to the electrical mesh architecture.

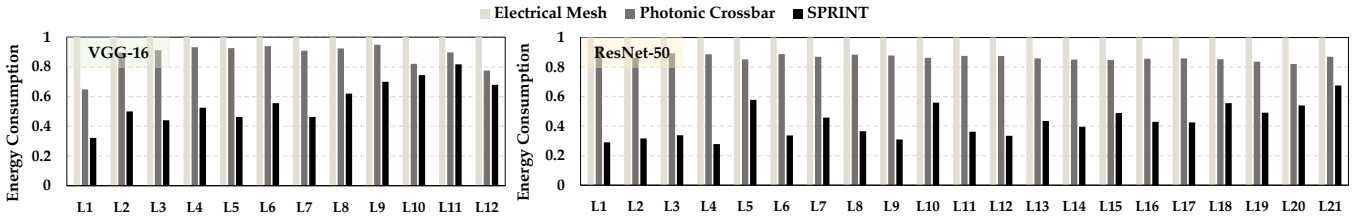



Fig. 11: Energy consumption comparison across different VGG-16 and ResNet-50 layers when utilizing WS accelerator chiplets. In each layer, all values are normalized to the electrical mesh architecture.

GLB is assumed to be evenly distributed to each accelerator chiplet when modeling electrical mesh and photonic crossbar inter-chiplet networks. By contrast, the GLB is assumed to be placed on a separate silicon die as shown in Fig. 5 when modeling the SPRINT architecture. To keep the laser power in a reasonable range, the maximal number of accelerator chiplets involved in broadcast communication is set to 16 in SPRINT architecture. In the case when the number of accelerator chiplets in the system exceed 16, broadcast communication from the GLB to all accelerator chiplets is implemented as several broadcast communications from the GLB to a subset of accelerator chiplets attached to the same GLB-to-Accelerator waveguide as shown in Fig. 5.

**Benchmarks**: We choose four CNN models, VGG-16 [35], ResNet-50 [32], DenseNet-201 [51], and EfficientNet-B7 [52] as the evaluation benchmarks. Table 3 lists the notations, layer names in `Caffe` [53], and the number of computations over the number of parameters (`Comp./Comm.`) of all 12 and 21 different layers in VGG-16 and ResNet-50, respectively. We will present the layer-by-layer simulation results of VGG-16 and ResNet-50 to closely examine the impact of layer parameters on system performance and energy consumption. For DenseNet-201 and EfficientNet-B7, we will only present the simulation results of a complete inference pass. Please note that we have removed redundant layers with the same configuration parameters. For example, res2a_branch1 in ResNet-50 has been removed because it has the same configuration parameters as res2[a-c]_branch2c (L4 in ResNet-50 in Table 3).

## 6 Simulation Results

### 6.1 Execution Time and Energy Consumption

**Execution Time**: Fig. 9 depicts the execution time comparison of SPRINT architecture and two other architectures, namely electrical mesh [54] and photonic crossbar, when using WS accelerator chiplets. As compared to the electrical mesh architecture, SPRINT achieves execution time reduction in the range of 27% (`L9:conv5-1`) to 76% (`L11:fc-4096`) and 28% (`L11:res4a_branch1`) to 66% (`L21:fc-4096`) in VGG-16 and ResNet-50, respectively. SPRINT performs extremely well in `L11:fc-4096` layer in VGG-16 due to the low $Comp./Comm.$ value of this layer, which means that the data is frequently moved around along the memory hierarchy. As compared to the photonic crossbar, SPRINT achieves execution time reduction in the range of 8% (`L9:conv5-1`) to 58% (`L11:fc-4096`) and 1% (`L11:res4a_branch1`) to 50% (`L21:fc-1000`) in VGG-16 and ResNet-50, respectively. The performance discrepancy of photonic crossbar and SPRINT architecture is relatively small, as they both exhibit distance-independent latency during data transmission. SPRINT architecture outperforms photonic crossbar because the wavelengths are allocated based on real bandwidth demand in SPRINT architecture, but equally allocated to chiplets in photonic crossbar.

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



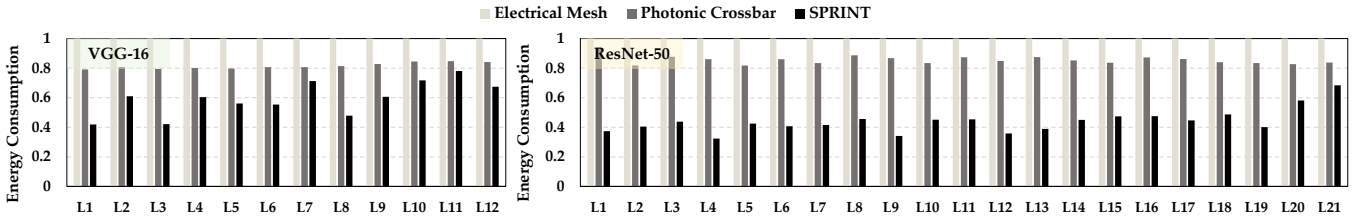



Fig. 12: Energy consumption comparison across different VGG-16 and ResNet-50 layers when utilizing RS accelerator chiplets. In each layer, all values are normalized to the electrical mesh architecture.

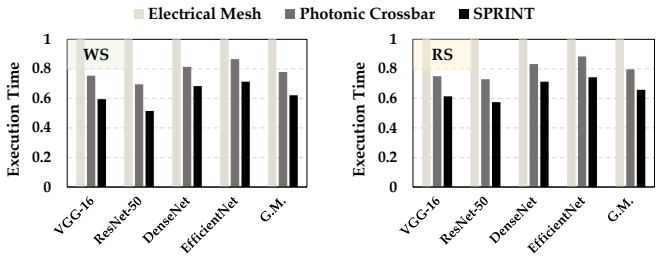



Fig. 13: Execution time comparison across VGG-16, ResNet-50, DenseNet-201, and EfficientNet-B7 when utilizing WS (left) and RS (right) accelerator chiplets. All values are normalized to the electrical mesh architecture.

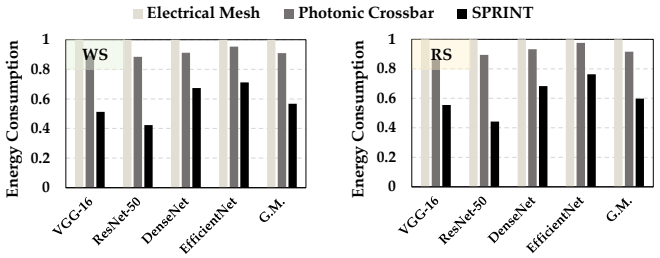



Fig. 14: Energy consumption comparison across VGG-16, ResNet-50, DenseNet-201, and EfficientNet-B7 when utilizing WS (left) and RS (right) accelerator chiplets. All values are normalized to the electrical mesh architecture.

Fig. 10 shows the execution time comparison of SPRINT architecture and two other architectures when using RS accelerator chiplets. As compared to the electrical mesh architecture, SPRINT achieves execution time reduction in the range of 28% (`L9:conv5-1`) to 63% (`L2:conv1-2`) and 24% (`L18:res5[a-c]_branch2b`) to 61% (`L5:res2[b-c]_branch2a`) in VGG-16 and ResNet-50, respectively. We make two observations here. First, the execution time reduction of SPRINT architecture over electrical mesh architecture is reduced when replacing WS chiplets with RS chiplets. This is because psums keep being streamed out of an accelerator chiplet in the original RS dataflow [22], possibly leading to more inter-chiplet communication. Second, different accelerator architectures and dataflows are suitable for layers with different configurations. As compared to the photonic crossbar, SPRINT achieves execution time reduction in the range of 14% (`L9:conv5-1`) to 39% (`L12:fc-4096`) and 11% (`L18:res5[a-c]_branch2b`) to 25% (`L5:res2[b-c]_branch2a`) in VGG-16 and ResNet-50, respectively. We can observe that the execution time reduction of SPRINT architecture over two other architectures vary when different chiplet types or different benchmarks are used.

**Energy Consumption**: Fig. 11 depicts the energy consumption comparison of SPRINT architecture and two other architectures when using WS accelerator chiplets. As compared to the electrical mesh architecture, SPRINT achieves energy consumption reduction in the range of 19% (`L11:fc-4096`) to 68% (`L1:conv1-1`) and 32% (`L21:fc-1000`) to 72% (`L4:res2[a-c]_branch2c`) in VGG-16 and ResNet-50, respectively. The reduction in energy consumption mainly comes from the energy saving in inter-chiplet communications. As compared to the photonic crossbar, SPRINT achieves energy consumption reduction in the range of 9% (`L10:fc-4096`) to 52% (`L3:conv2-1`) and 22% (`L21:fc-1000`) to 69% (`L1:conv1`) in VGG-16 and ResNet-50, respectively. As compared to photonic crossbar, the reduction in energy consumption in SPRINT architecture comes from fewer MRRs required and their peripheral circuitries. Fig. 12 shows the energy consumption comparison when RS accelerator chiplets are used. We observe similar trend as in Fig. 11.

Since DenseNet-201 and EfficientNet-B7 includes many layers, it's difficult to present the per-layer execution time and energy consumption when comparing SPRINT with electrical mesh and photonic crossbar. We only present the execution time and energy consumption of one complete inference pass using four CNN models on WS and RS accelerators in Fig. 13 and Fig. 14. We observe that though SPRINT achieves the most reduction in execution time and energy consumption in ResNet-50 and VGG-16, respectively, SPRINT performs well in the more recent DenseNet-201 and EfficientNet-B7 models as well.

### 6.2 SPRINT Adaptability

We study whether the SPRINT photonic inter-chiplet network can adapt to different chiplet-level architectures and dataflows using four different types of chiplets and corresponding dataflows listed in Table 1. Simulation results in Fig. 15 and Fig. 16 show that SPRINT outperforms the other two architectures, in terms of execution time (up to 55% reduction) and energy consumption (up to 52% reduction), when all four types of chiplets are used, indicating the superior adaptability of SPRINT photonic inter-chiplet network. Specifically, SPRINT architecture achieves the most significant execution time and energy consumption reduction when NLR chiplets are used, as the large amount of inter-chiplet data transmission incurred in NLR dataflow

This article has been accepted for publication in IEEE Transactions on Parallel and Distributed Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2021.3139015



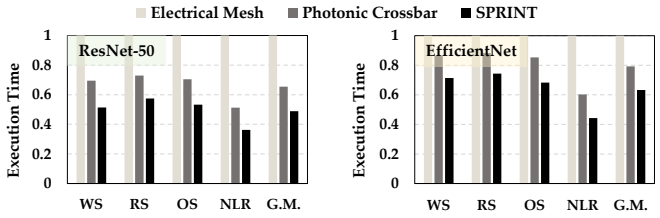



Fig. 15: Execution time comparison across ResNet-50 (left) and EfficientNet-B7 (right) when utilizing WS, RS, OS, and NLR accelerator chiplets. All values are normalized to the electrical mesh architecture.

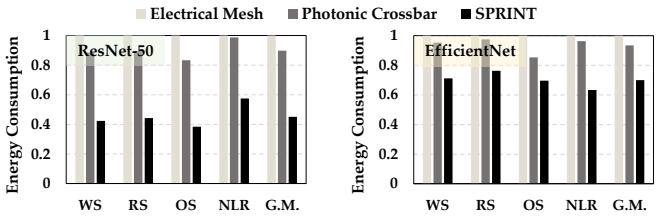



Fig. 16: Energy consumption comparison across ResNet-50 (left) and EfficientNet-B7 (right) when utilizing WS, RS, OS, and NLR accelerator chiplets. All values are normalized the electrical mesh architecture.

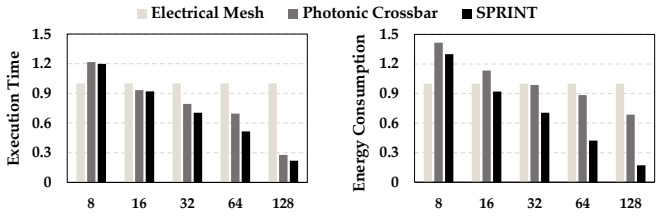



Fig. 17: Execution time and energy consumption comparison when varying the number of accelerator chiplets.

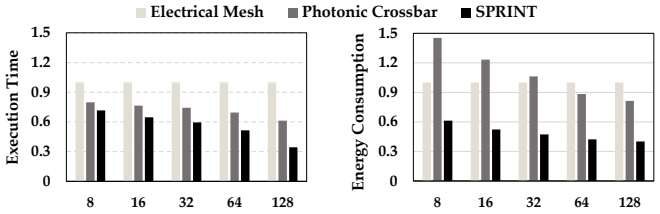



Fig. 18: Execution time and energy consumption comparison when varying the number PEs per accelerator chiplet.

can fully exploit the benefit of SPRINT photonic inter-chiplet network. SPRINT architecture achieves the least execution time and energy consumption reduction when RS chiplets are used, as the enhanced intra-chiplet data reuse in RS dataflow makes the impact of implementing an advanced inter-chiplet network less significant.

### 6.3 SPRINT Scalability

We study the scalability of the SPRINT architecture by varying the number of accelerator chiplets in the system from 8 to 128. Fig. 17 shows the comparisons of execution time and energy consumption. When integrating 128 accelerator chiplets in the system, SPRINT architecture achieves up to 78% and 83% reduction in execution time and energy consumption, respectively, as compared to other architectures. We make the following observations: (1) photonic crossbar and SPRINT architecture perform worse than electrical mesh when the number of chiplets is low because of the expensive E/O and O/E signal conversions (2) photonic crossbar and SPRINT architecture show good scalability in terms of execution time as the system scale increases because of the distance-independent property of photonics; (3) the energy consumption of photonic crossbar, though scales better than electrical mesh, is much higher than that of SPRINT architecture due to the large number of MRRs required; (4) we can project good scalability of SPRINT when the number of chiplets increases beyond 128. We also study the scalability of the SPRINT architecture by varying the number of PEs per accelerator chiplet from 8 to 128 while keeping the number of accelerator chiplets at 64. Fig. 18 shows that SPRINT performs better in terms of execution time and energy consumption as the number of PEs per accelerator chiplet increases.

### 6.4 Implementation Cost

To achieve equal per-chiplet bandwidth, SPRINT architecture requires 14.5 $K$ MRRs while photonic crossbar requires over 338 $K$ MRRs, which is 23 times larger. This is because the number of MRRs scales linearly and quadratically with the number of chiplets in SPRINT architecture and photonic crossbar, respectively. [55] has reported a 6-bit DAC with 29 $mW$ power consumption (130 $nm$ technology). We assume that a 6-bit DAC using 28 $nm$ technology consumes 2.7 $mW$ power using the scaling factor provided in [50]. Since all the tunable splitters attached to a chiplet share a split ratio, we implement one 6-bit DAC in the reconfiguration controller unit (RCU) and it incurs insignificant area and power overhead. Fig. 19 shows that the energy consumption of DACs is only 1.2% to 1.7% of the overall energy consumption of the SPRINT photonic inter-chiplet network. As for timing overhead, the switching time for a DAC is 200 $ps$ [37] while the tuning time for a tunable splitter is 500 $ps$ [37]. In our evaluation, we assume that a new split ratio can be set within 1 $ns$.

## 7 Related Work

**CNN Accelerator**: Several CNN accelerators with different dataflows [20], [22], [23], [26], [27], [56] have been proposed in recent years. These accelerators are implemented on monolithic chips and focus on improving data reuse [21] assuming uniform latency and bandwidth across PEs. However, the increasing CNN model size requires scale-up systems like chiplet-based architectures, in which latency between PEs in different chiplets becomes significant and the uniform latency and bandwidth assumption no longer holds true. Very few prior work [5], [6], [57] explores implementing machine learning models in chiplet-based architectures. Shao *et al.* [5] implement the CNN on a chiplet-based architecture with an electrical mesh network for inter-chiplet communication. Though this work adopts aggressive electrical wire technology to implement the inter-chiplet network, it still becomes the performance bottleneck as the system scales up, indicating the scalability limitation of electrical networks at the chiplet scale. Hwang *et al.*

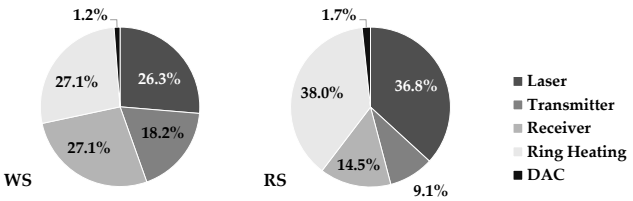



Fig. 19: Energy consumption breakdown of SPRINT photonic network when utilizing WS and RS accelerator chiplets.

[57] implement a recommendation model on a chiplet-based architecture. This work focuses on the accelerator design and the interconnection network between chiplets is an electrical bus. Ascia et al. [6] propose to replace the electrical wires with wireless channels to implement the inter-chiplet network for CNN inference. Though exhibiting higher scalability as compared to the electrical networks, the cost of providing sufficient bandwidth is significant. Our work explores scaling up the chiplet-based architecture using photonic interconnects for large CNN models, as photonic interconnects are expected to achieve higher scalability and bandwidth density.

**Photonic Interconnects in Chiplet-based Architectures**: Implementing photonic interconnects on chip has been well studied [38], [58], [59], [60], [61], [62]. Prior work [12], [14], [15], [16], [17], [63], [64], [65] has explored to implement photonic interconnects on silicon interposer in chiplet-based architectures. Demir *et al.* [14] propose to construct a many-core "virtual chip" by connecting multiple smaller chiplets through a photonic crossbar. Photonic crossbars are also used to connect the chiplets in [16], [18]. Grani *et al.* [15] utilize arrayed waveguide grating router (AWGR) photonic interconnects implemented on the silicon interposer to realize a 16×16 photonic network-on-chip (NoC). Prior work leverages the distance-independent feature of photonics in large-scale chiplet-based architectures. However, the implementation cost is often high as all-to-all communication is assumed and bandwidth resources are uniformly distributed. Our work explores to exploit the specific communication patterns in CNN inference to further improve the performance and reduce the implementation cost of the inter-chiplet interconnects.

## 8 Conclusion

In this paper, we present a chiplet-based CNN accelerator named SPRINT. The unique features of SPRINT include (1) a novel photonic inter-chiplet network that is optimized to adapt to specific communication patterns in CNN inference through wavelength allocation and waveguide reconfiguration, and (2) a novel optically-enhanced CNN dataflow that exploits the inherent broadcasting and multicasting capabilities of photonics to efficiently support the prevalent broadcasting communication in CNN inference. Simulation studies using multiple CNN models show that SPRINT provides significant reduction in execution time and energy consumption, as well as superior scalability, as compared to other state-of-the-art chiplet-based architectures with metallic or photonic inter-chiplet interconnects.

## 9 Acknowledgments

This research was partially supported by National Science Foundation grants CCF-1702980, CCF-1812495, CCF-1901165, CCF-1953980, CCF-1513606, CCF-1703013, and CCF-1901192. We sincerely thank the anonymous reviewers for the excellent feedback.

## References

[1] R. Mayer and H. A. Jacobsen. Scalable Deep Learning on Distributed Infrastructures: Challenges, Techniques, and Tools. *ACM Computing Survey*, 53(1):1–37, 2020.

[2] V. Sze, Y. Chen, T. Yang, and J. S. Emer. Efficient Processing of Deep Neural Networks: A Tutorial and Survey. *Proceedings of the IEEE*, 105(12):2295–2329, 2017.

[3] B. Klenk and L. Dennison. Why Data Science and Machine Learning Need Silicon Photonics. In *Optical Fiber Communication Conference*, pages M4F–6, 2020.

[4] S. M. Nabavinejad, M. Baharloo, K. Chen, M. Palesi, T. Kogel, and M. Ebrahimi. An Overview of Efficient Interconnection Networks for Deep Neural Network Accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 10(3):268–282, 2020.

[5] 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, C. T. Gray, B. Khailany, and S. W. Keckler. Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, page 14–27, 2019.

[6] G. Ascia, V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. Improving Inference Latency and Energy of DNNs through Wireless Enabled Multi-Chip-Module-Based Architectures and Model Parameters Compression. In *Proceedings of the IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pages 1–6, 2020.

[7] A. Kannan, N. E. Jerger, and G. H. Loh. Enabling Interposer-Based Disintegration of Multi-Core Processors. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 546–558, 2015.

[8] H. Zheng, K. Wang, and A. Louri. A Versatile and Flexible Chiplet-Based System Design for Heterogeneous Manycore Architectures. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2020.

[9] D. A. B. Miller. Device Requirements for Optical Interconnects to Silicon Chips. *Proceedings of the IEEE*, 97(7):1166–1185, 2009.

[10] 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)*, pages 1–6, 2021.

[11] S. Van Winkle, A. Karanth, R. Bunescu, and A. Louri. Extending the Power-Efficiency and Performance of Photonic Interconnects for Heterogeneous Multicores with Machine Learning. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 480–491, 2018.

[12] A. Narayan, Y. Thonnart, P. Vivet, and A. K. Coskun. PROWAVES: Proactive Runtime Wavelength Selection for Energy-Efficient Photonic NoCs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–14, 2020.

[13] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. Emer, and D. Englund. Freely Scalable and Reconfigurable Optical Hardware for Deep Learning. *Scientific Reports*, 11(1):1–12, 2021.

[14] 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)*, page 303–312, 2014.

[15] P. Grani, R. Proietti, V. Akella, and S. J. Ben 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)*, pages 289–300, 2017.

[16] A. Narayan, Y. Thonnart, P. Vivet, C. F. Tortolero, and A. K. Coskun. WAVES: Wavelength Selection for Power-Efficient 2.5D-Integrated Photonic NoCs. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, pages 516–521, 2019.

[17] P. Fotouhi, S. Werner, J. Lowe-Power, and S. J. Ben Yoo. Enabling Scalable Chiplet-Based Uniform Memory Architectures with Silicon Photonics. In *Proceedings of the International Symposium on Memory Systems (MEMSYS)*, page 222–334, 2019.

[18] Y. Thonnart, S. Bernabé, 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 and Test in Europe Conference (DATE)*, pages 1456–1461, 2020.

[19] A. Narayan, Y. Thonnart, P. Vivet, A. Joshi, and A. K. Coskun. System-Level Evaluation of Chip-Scale Silicon Photonic Networks for Emerging Data-Intensive Applications. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*, pages 1444–1449, 2020.

[20] H. Kwon, A. Samajdar, and T. Krishna. MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects. In *Proceedings of the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, page 461–475, 2018.

[21] 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 *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, page 754–768, 2019.

[22] Y. 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)*, page 367–379, 2016.

[23] 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)*, page 92–104, 2015.

[24] 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)*, page 247–257, 2010.

[25] 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)*, page 199–204, 2015.

[26] H. J. Yoo, S. Park, K. Bong, D. Shin, J. Lee, and S. Choi. A 1.93 TOPS/W Scalable Deep Learning/Inference Processor with Tetra-Parallel MIMD Architecture for Big Data Applications. In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 80–81, 2015.

[27] 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)*, page 269–284, 2014.

[28] 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*, 97(7):1337–1361, 2009.

[29] M. Petracca, B. G. Lee, K. Bergman, and L. P. Carloni. Design Exploration of Optical Interconnection Networks for Chip Multiprocessors. In *IEEE Symposium on High Performance Interconnects*, pages 31–40, 2008.

[30] R. G. Beausoleil, J. Ahn, N. Binkert, A. Davis, D. Fattal, M. Fiorentino, N. P. Jouppi, M. McLaren, C. M. Santori, R. S. Schreiber, S. M. Spillane, D. Vantrease, and Q. Xu. A Nanophotonic Interconnect for High-Performance Many-Core Computation. In *IEEE Symposium on High Performance Interconnects*, pages 182–189, 2008.

[31] 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)*, pages 153–164, 2008.

[32] 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)*, pages 770–778, 2016.

[33] 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)*, pages 1–9, 2015.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[35] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, pages 1–14, 2014.

[36] A. Biberman, H. L. R. Lira, K. Padmaraju, N. Ophir, J. Chan, M. Lipson, and K. Bergman. Broadband Silicon Photonic Electrooptic Switch for Photonic Interconnection Networks. *IEEE Photonics Technology Letters (PTL)*, 23(8):504–506, 2011.

[37] E. Peter, A. Thomas, A. Dhawan, and S. R. Sarangi. Active Microring based Tunable Optical Power Splitters. *Optics Communications*, 359:311–315, 2016.

[38] J. Bashir, E. Peter, and S. R. Sarangi. A Survey of On-Chip Optical Interconnects. *ACM Computing Survey*, 51(6):1–34, 2019.

[39] X. Hu, D. Stow, and Y. Xie. Die Stacking Is Happening. *IEEE Micro*, 38(1):22–28, 2018.

[40] H. Venghaus. *Wavelength Filters in Fibre Optics*. Springer, 2006.

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

[42] 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)*, pages 282–293, 2012.

[43] C. Sun, C. 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)*, pages 201–210, 2012.

[44] S. Werner, J. Navaridas, and M. Luján. 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)*, pages 265–276, 2017.

[45] R. 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)*, pages 207–214, 2010.

[46] 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)*, pages 124–133, 2009.

[47] P. Rosenfeld, E. Cooper-Balis, and B. Jacob. DRAMSim2: A Cycle Accurate Memory System Simulator. *IEEE Computer Architecture Letters (CAL)*, 10(1):16–19, 2011.

[48] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi. CACTI 6.0: A Tool to Model Large Caches. *HP laboratories*, 27:1–24, 2009.

[49] R. Polster, Y. Thonnart, G. Waltener, J. Gonzalez, and E. Cassan. Efficiency Optimization of Silicon Photonic Links in 65-nm CMOS and 28-nm FDSOI Technology Nodes. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(12):3450–3459, 2016.

[50] A. Stillmaker and B. Baas. Scaling Equations for the Accurate Prediction of CMOS Device Performance from 180 nm to 7 nm. *Integration*, 58:74–81, 2017.

[51] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4700–4708, 2017.

[52] M. Tan and Q. V. Le. Efficientnet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[53] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the ACM International Conference on Multimedia*, page 675–678, 2014.

[54] Y. Li and A. Louri. ALPHA: A Learning-Enabled High-Performance Network-on-Chip Router Design for Heterogeneous Manycore Architectures. *IEEE Transactions on Sustainable Computing*, 6(2):274–288, 2021.

[55] X. Wu, P. Palmers, and M. S. J. Steyaert. A 130 nm CMOS 6-Bit Full Nyquist 3 GS/s DAC. *IEEE Journal of Solid-State Circuits*, 43(11):2396–2403, 2008.

[56] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam. DaDianNao: A Machine-Learning Supercomputer. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture (MICRO)*, page 609–622, 2014.

[57] R. Hwang, T. Kim, Y. Kwon, and M. Rhu. Centaur: A Chiplet-Based, Hybrid Sparse-Dense Accelerator for Personalized Recommendations. In *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*, page 968–981, 2020.

[58] R. R. Tummala. Moore's Law Meets its Match (System-on-Package). *IEEE Spectrum*, 43(6):44–49, 2006.

[59] S. Werner, J. Navaridas, and M. Luján. A Survey on Optical Network-on-Chip Architectures. *ACM Computing Survey*, 50(6):1–37, 2017.

[60] T. Alexoudi, N. Terzenidis, S. Pitris, M. Moralis-Pegios, P. Maniotis, C. Vagionas, C. Mitsolidou, G. Mourgias-Alexandris, G. T. Kanellos, A. Miliou, K. Vyrsokinos, and N. Pleros. Optics in Computing: From Photonic Network-on-Chip to Chip-to-Chip Interconnects and Disintegrated Architectures. *Journal of Lightwave Technology*, 37(2):363–379, 2019.

[61] Y. Demir and N. Hardavellas. SLaC: Stage Laser Control for a Flattened Butterfly Network. In *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 321–332, 2016.

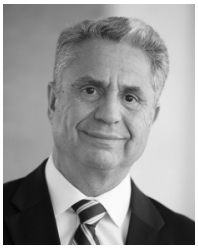
[62] M. Wade, E. Anderson, S. Ardalan, P. Bhargava, S. Buchbinder, M. L. Davenport, J. Fini, H. Lu, C. Li, R. Meade, C. Ramamurthy, M. Rust, F. Sedgwick, V. Stojanovic, D. Van Orden, C. Zhang, C. Sun, S. Y. Shumarayev, C. O'Keeffe, T. T. Hoang, D. Kehlet, R. V. Mahajan, M. T. Guzy, A. Chan, and T. Tran. TeraPHY: A Chiplet Technology for Low-Power, High-Bandwidth In-Package Optical I/O. *IEEE Micro*, 40(2):63–71, 2020.

[63] 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)*, pages 1–13, 2022.

[64] A. Coskun, F. Eris, A. Joshi, A. B. Kahng, Y. Ma, A. Narayan, and V. Srinivas. Cross-Layer Co-Optimization of Network Design and Chiplet Placement in 2.5-D Systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):5183–5196, 2020.

[65] Z. Wang, Z. Wang, J. Xu, Y. S. Chang, J. Feng, X. Chen, S. Chen, and J. Zhang. CAMON: Low-Cost Silicon Photonic Chiplet for Manycore Processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(9):1820–1833, 2020.


**Yuan Li** received the BS degree in physics from the University of Science and Technology of China in 2010, and the MS degree in microelectronics from the University of Newcastle upon Tyne in 2011. He is currently working toward the PhD degree in computer engineering at the George Washington University. His research interests include machine learning architectures, accelerator-rich heterogeneous systems, and emerging interconnect and memory technologies. He is a student member of the IEEE.



**Ahmed Louri** is the David and Marilyn Karlgaard Endowed Chair Professor of Electrical and Computer Engineering at the George Washington University, which he joined in August 2015. He is also the director of the High Performance Computing Architectures and Technologies Laboratory. Dr. Louri received the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles, California in 1988. From 1988 to 2015, he was a professor of Electrical and Computer Engineering at the University of Arizona, and during that time, he served six years (2000 to 2006) as the Chair of the Computer Engineering Program. From 2010 to 2013, Dr. Louri served as a program director in the National Science Foundation's (NSF) Directorate for Computer and Information Science and Engineering. He directed the core computer architecture program and was on the management team of several cross-cutting programs. Dr. Louri conducts research in the broad area of computer architecture and parallel computing, with emphasis on interconnection networks, optical interconnects for scalable parallel computing systems, reconfigurable computing systems, and power-efficient and reliable Network-on-Chips (NoCs) for multicore architectures. Recently he has been concentrating on: energy-efficient, reliable, and high-performance many-core architectures; accelerator-rich reconfigurable heterogeneous architectures; machine learning techniques for efficient computing, memory, and interconnect systems; emerging interconnect technologies (photonic, wireless, RF, hybrid) for NoCs; future parallel computing models and architectures (including convolutional neural networks, deep neural networks, and approximate computing); and cloud-computing and data centers. He is the recipient of the 2020 IEEE Computer Society Edward J. McCluskey Technical Achievement Award, "for pioneering contributions to the solution of on-chip and off-chip communication problems for parallel computing and manycore architectures." Dr. Louri is a Fellow of the IEEE, and he is currently the Editor-in-Chief of the IEEE Transactions on Computers. More information can be found at https://hpcat.seas.gwu.edu/Director.html.

**Avinash Karanth** received the BE degree in electronics and communications in February 2000 from the Manipal Institute of Technology, Mangalore University, and the MS and PhD degrees in the Electrical and Computer Engineering Department from The University of Arizona in May 2003 and August 2006, respectively. Presently, he is the Joseph Jachinowski Professor in the School of Electrical Engineering and Computer Science at Ohio University in Athens, Ohio. Dr. Karanth directs the Technologies for Emerging Computer Architecture Lab (TEAL) at Ohio University. His research interests include computer architecture, optical interconnects, Network-on-Chips (NoCs) and emerging technologies such as nanophotonics, 3D, and wireless interconnects. He is the recipient of the NSF CAREER Award in 2011, the Presidential Research Scholar Award in 2017, the Best Paper Award at the ICCD 2013 conference and his papers have been nominated for Best Paper at the IEEE Symposium on Network-on-Chips (NoCs) in May 2010 and the IEEE Asia & South Pacific Design Automation Conference (ASP-DAC) in January 2009. He is a senior member of the IEEE and member of ACM.