SIAM: Chiplet-based Scalable In-Memory Acceleration with Mesh for Deep Neural Networks

GOKUL KRISHNAN, Arizona State University, USA
SUMIT K. MANDAL, University of Wisconsin-Madison, USA
MANVITHA PANNALA, CHAITALI CHAKRABARTI, and JAE-SUN SEO, Arizona State University, USA
UMIT Y. OGRAS, University of Wisconsin-Madison, USA
YU CAO, Arizona State University, USA

In-memory computing (IMC) on a monolithic chip for deep learning faces dramatic challenges on area, yield, and on-chip interconnection cost due to the ever-increasing model sizes. 2.5D integration or chiplet-based architectures interconnect multiple small chips (i.e., chiplets) to form a large computing system, presenting a feasible solution beyond a monolithic IMC architecture to accelerate large deep learning models. This paper presents a new benchmarking simulator, SIAM, to evaluate the performance of chiplet-based IMC architectures and explore the potential of such a paradigm shift in IMC architecture design. SIAM integrates device, circuit, architecture, network-on-chip (NoC), network-on-package (NoP), and DRAM access models to realize an end-to-end system. SIAM is scalable in its support of a wide range of deep neural networks (DNNs), customizable to various network structures and configurations, and capable of efficient design space exploration. We demonstrate the flexibility, scalability, and simulation speed of SIAM by benchmarking different state-of-the-art DNNs with CIFAR-10, CIFAR-100, and ImageNet datasets. We further calibrate the simulation results with a published silicon result, SIMBA. The chiplet-based IMC architecture obtained through SIAM shows 130× and 72× improvement in energy-efficiency for ResNet-50 on the ImageNet dataset compared to Nvidia V100 and T4 GPUs.

CCS Concepts: • **Hardware → Analysis and Design of Emerging Devices and Systems**; **On-chip Resource Management**; **Emerging Architectures**; **Interconnect**; **System-on-a-chip;**

Additional Key Words and Phrases: Chiplet architecture, in-memory compute, DNN acceleration, IMC benchmarking, network-on-chip, network-on-package

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2021.
This work was supported by C-BRIC, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and SRC GRC Task 3012.001.
Authors' addresses: G. Krishnan, M. Pannala, C. Chakrabarti, J.-s. Seo, and Y. Cao, Arizona State University, School of Electrical, Computer, and Energy Engineering, Tempe, AZ, 85287, USA; emails: {gkrish19, mpannal1, chaitali, jseo28, Yu.Cao}@asu.edu; S. K. Mandal and U. Y. Ogras, University of Wisconsin-Madison, Department of Electrical and Computer Engineering, Madison, WI, 53706; emails: {skmandal, uogras}@wisc.edu.
Updated author affiliation: GOKUL KRISHNAN, Arizona State University, USA; SUMIT K. MANDAL, University of Wisconsin-Madison, USA; MANVITHA PANNALA, Arizona State University, USA; CHAITALI CHAKRABARTI, Arizona State University, USA; JAE-SUN SEO, Arizona State University, USA; UMIT Y. OGRAS, University of Wisconsin-Madison, USA; YU CAO, Arizona State University, USA.

1539-9087/2021/09-ART68 $15.00
https://doi.org/10.1145/3476999

ACM Reference format:
Gokul Krishnan, Sumit K. Mandal, Manvitha Pannala, Chaitali Chakrabarti, Jae-sun Seo, Umit Y. Ogras, and Yu Cao. 2021. SIAM: Chiplet-based Scalable In-Memory Acceleration with Mesh for Deep Neural Networks. *ACM Trans. Embedd. Comput. Syst.* 20, 5s, Article 68 (September 2021), 24 pages.
https://doi.org/10.1145/3476999

1 INTRODUCTION

State-of-the-art deep neural networks (DNNs) have become more complex with wider, deeper, and more branched structures to cater to the needs of various applications [11, 12]. For instance, network architecture search (NAS) methods generate highly branched and complex DNNs, which increase compute and memory requirements [41, 46]. In-memory computing (IMC)-based architectures can support these network models because of their ability to embed deep learning operations in the memory array, achieving massively parallel computing with high storage density. Prior studies demonstrated that crossbar-based IMC architectures with RRAM or SRAM significantly improved throughput and energy-efficiency for DNN accelerators [14, 20, 23, 34, 38]. Such architectures usually assume all DNN weights are stored on a monolithic chip to minimize DRAM access and maximize parallel IMC computing. However, as the DNN model size becomes larger and larger, this approach leads to increased chip area and on-chip memory.

Figure 1(a) shows the total chip area for a monolithic RRAM-based IMC architecture across different DNNs [20]. Larger and branched DNNs like DenseNet-110 [12] result in a chip area of up to $1,200\text{mm}^2$. The increased area is attributed to the larger model size and the branched structure in DNNs. For example, ResNet-50 has 23M parameters and residual connections (branched connections). For an 8-bit precision, the mapping scheme in [34], and a crossbar size of 128x128, results in 802 tiles. Here each tile consists of 16 IMC crossbar arrays. In addition, the presence of the residual connections results in increased buffer cost due to the need to store the activations of previous layers to perform residual addition operations in the ResNet DNN. For the same hardware configuration, LeNet-5 with 0.43M parameters requires 43 tiles, while DenseNet-110 with 28.1M parameters requires 2184 tiles for the same hardware configuration. Hence, the DNN size and structure influence the overall area and, in turn, fabrication cost. Furthermore, higher chip area further causes lower yield and higher defects across the wafer [18], resulting in wasted area and a higher fabrication cost. Figure 1(a) also shows the fabrication cost of the monolithic RRAM-based IMC architecture for different DNNs. We see that the fabrication cost increases exponentially with increased chip area (note that the cost is shown in the logarithmic scale), thus making the monolithic IMC architecture much less cost-efficient if all weights are stored on a single chip. Hence, there is an urgent need to address the increased fabrication cost of IMC-based DNN accelerators [20, 34, 38].

2.5D integration or chiplet-based architectures [3, 22, 35, 45] provide a promising alternative to monolithic hardware architectures. They integrate multiple chiplets through silicon interposers or organic substrates [9, 40]. Compared to the monolithic chip, the smaller chiplet size helps improve the design effort, yield, reduces defect ratio, and reduces fabrication cost. Figure 1(b) shows a representative 3-dimensional diagram of a chiplet-based IMC architecture. Chiplets comprise of memory units, computation blocks, and DRAM allowing for large-scale system integration. The interposer acts as an additional routing layer (network-on-package or NoP) that utilizes package-level signaling to connect different chiplets. Recent advances in package-level signaling have enabled a 2× improvement in bandwidth over board-level interconnections with 8× lower energy-per-bit [22, 40].

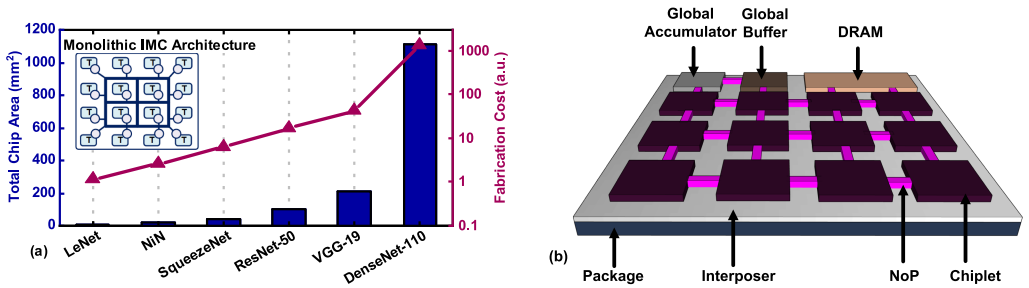

Fig. 1. (a) Total chip area and fabrication cost for a monolithic RRAM-based IMC architecture for different DNNs [34]. Fabrication cost increases exponentially with an increase in total chip area (Appendix A details the method to calculate the fabrication cost). (b) 3-dimensional diagram showing the chiplet-based IMC architecture. The architecture includes an array of IMC chiplets, global buffer, global accumulator, and DRAM connected by an NoP. The figure is for representational purposes and not drawn to scale.

In a chiplet-based architecture, the design space parameters primarily include IMC crossbar size, the number of crossbars per chiplet, the number of chiplets, network-on-chip (NoC), NoP, and the DNN structure. These parameters need to be optimized systematically in order to exploit the potential provided by this new architecture. For example, SIMBA [35], a chiplet-based accelerator developed by Nvidia, utilizes 36 chiplets, with each chiplet consisting of 16 processing elements (PEs) [35]. While the underlying architecture ensures correct general-purpose functionality, it does not guarantee optimal performance for various DNN applications. Therefore, an extensive design space exploration is required to identify the optimal chiplet-based IMC architecture for DNN inference.

In this work, we propose a novel chiplet-based IMC architecture simulator, SIAM, that integrates device, circuits, architecture, NoC, NoP, and DRAM access estimation under a single roof for design space exploration. We plan to open-source SIAM upon acceptance of this work. To the best of our knowledge, SIAM will be the first open-sourced chiplet-based IMC architecture simulator to promote architectural research in this emerging domain. SIAM includes four main components: partition and mapping engine, circuit and NoC engine, NoP engine, and DRAM engine. A Python wrapper is used to interface each engine with one another. The wrapper also interfaces SIAM with popular deep learning frameworks such as TensorFlow and PyTorch.

SIAM provides a scalable solution that utilizes model-based and cycle-accurate simulation components, allowing for performance evaluation of a wide range of DNNs across multiple datasets. It has a flexible architecture to support multiple DNN to IMC chiplet and crossbar partition and mapping schemes, thus generating different types of chiplet-based IMC architectures. Thus, SIAM provides a platform to enable comparisons across different chiplet-based IMC architectures and also between chiplet-based and monolithic IMC architectures. Furthermore, SIAM has a low simulation time to support the fast design and benchmarking exploration. For example, ResNet-110 with 1.7M parameters takes 12 minutes, while VGG-16 with 138M parameters takes 4.26 hours for benchmarking.

We demonstrate SIAM's capabilities by conducting experiments on state-of-the-art DNNs such as ResNet-110 [10] for CIFAR-10, VGG-19 [36] for CIFAR-100, and ResNet-50 [10] and VGG-16 [36] for ImageNet datasets. Furthermore, to evaluate SIAM at the system level , we calibrate SIAM against a published silicon result, SIMBA [35], especially the scaling trend with the number of chiplets. The major contributions of this work are three-fold:

- We propose a complete framework, SIAM, that combines IMC circuit, NoC, NoP, and DRAM performance evaluation under a single roof. *SIAM is the first simulator to provide support for hardware performance evaluation of chiplet-based IMC architectures.* We carefully model the components of architecture like the IMC circuit, NoC, network-on-package (NoP), and DRAM.
- We provide a high degree of freedom to the user through different mapping schemes and customizable architectural parameters for IMC circuit, NoC, NoP, and DRAM components. We demonstrate different architectural design space exploration experiments that can be performed using SIAM.
- Extensive experimental evaluation of the SIAM simulator for different DNNs across CIFAR-10, CIFAR-100, and ImageNet datasets. Furthermore, we perform detailed experiments to calibrate the simulator to a real-design, SIMBA [35], making SIAM a reliable and accurate simulator. For ResNet-50 on ImageNet, the generated chiplet-based IMC architecture achieves 130× and 72× improvement in energy-efficiency compared to Nvidia V100 and T4 GPUs.

2 RELATED WORK

2.1 In-Memory Computing

In-memory computing is a promising alternative to conventional von-Neumann architectures [14, 34, 43]. Prior works mostly focus on a monolithic IMC-based architecture with all weights on-chip to minimize the access to external memory. Such an assumption leads to a higher area cost and, in turn, a higher fabrication cost, as shown in Figure 1(a). Higher area cost leads to lower yield and other difficulties in fabrication. To address this, in this work, we develop a benchmarking tool, SIAM, that utilizes a chiplet-based IMC architecture for DNN inference. Each chiplet integrates an array of IMC crossbars and associated peripheral circuits to perform the MAC operations. The chiplets also consist of buffers, accumulator circuits, pooling units, non-linear activation units, NoP driver circuit, and NoP router all interconnected using an NoC [25]. It consumes less per-chip area than a monolithic IMC architecture and results in lower fabrication cost and higher yield while providing similar system-level inference performance as the monolithic IMC architecture.

2.2 Chiplet-based Architectures

2.5D or chiplet-based architectures utilize package-level integration of small blocks, known as chiplets. Chiplet-based architectures provide a promising solution to build cost-effective large-scale systems for complex applications such as DNN inference. Chiplets consist of computation and memory units that form the primary building blocks of any architecture. Multiple chiplets are connected using an on-package network integrated into a silicon interposer or an organic substrate. Prior work has shown multiple chiplet-based architectures for various applications such as DNN acceleration, general purpose system-on-chips (SoCs), and recommendation systems [3, 7, 13, 22, 35, 42, 45], which present different on-package signaling techniques and associated circuitries. We limit the description of these works to three for brevity. A fine-grained chiplet architecture for deep learning inference is proposed in [35]. A custom NoP driver and associated interconnect that provides improved performance for large-scale DNN inference is utilized. A high performance SoC with up to 8 CPU cores that can scale across different market segments (edge and cloud) is proposed in [3]. A custom infinity fabric (NoP) that utilizes a physical layer for in-die and across package communication is utilized. Finally, [13] proposed a chiplet-based hybrid sparse-dense accelerator that utilizes a package-integrated CPU+FPGA system for personalized recommendation applications. A sparse accelerator is used for high-throughput embedding gather and reduction operations, while a dense accelerator is used for accelerating the compute-intensive DNN

Table 1. Comparison between Different IMC Simulators

Simulator	Architecture	Circuit	Interconnect	NoP Interconnect	DRAM
GenieX [6]	Monolithic	SPICE-based	No	No	No
RxNN [15]	Monolithic	SPICE-based	No	No	No
NeuroSim [29]	Monolithic	SPICE-based	P2P (H-Tree)	No	No
MNSIM [44]	Monolithic	Behavior model	NoC-mesh	No	No
SIAM	**Monolithic & Chiplet**	**SPICE and Behavioral Model**	**NoC-mesh, NoC-tree, and H-Tree**	**Supported (driver and interconnect)**	**Supported**

layers. However, all these prior studies focus on custom designs for specific applications, leaving little room for architecture-level benchmarking and design space exploration for chiplet-based architectures. Furthermore, they focus on CMOS-based conventional von-Neumann architectures, with little or no focus on IMC architectures. To address this, SIAM provides a benchmarking tool for IMC architectures based on chiplet structures with design space exploration capabilities.

2.3 Benchmarking Tools

Benchmarking tools enable a fast design space exploration. Prior work have proposed a number of benchmarking tools for IMC-based architectures [2, 6, 15, 21, 29, 32, 44] and conventional von-Neumann architectures [4, 31, 33]. Authors in [29] provide a DNN inference benchmarking tool for different device technologies. The architecture utilizes an array of IMC crossbars and peripheral circuits connected by point-to-point (P2P) interconnect for on-chip data movement. Recently, authors in [44] proposed a behavioral performance benchmarking tool for IMC architectures. They utilize individual models for each component in the IMC architecture to perform the estimation. All the prior simulators suffer from two significant drawbacks. *First*, they assume a custom monolithic IMC architecture for a given DNN. With a custom design, the architecture and resource utilization are different for each DNN, i.e., for each network, a specific IMC architecture is generated and evaluated to provide the desired inference hardware performance. Such an assumption leads to a need for designing an IMC architecture that is specific to a DNN. *Second,* the authors provide support only for monolithic IMC architectures, with no support for chiplet-based IMC structures.

In contrast to all this research, this work proposes a novel, general-purpose chiplet-based IMC architecture simulator, SIAM, that integrates device, circuits, architecture, NoC, NoP, and DRAM access estimation for design space exploration. SIAM utilizes smaller individual chiplets connected by the NoP fabric. Smaller chiplets lead to a lower area and fabrication cost. Furthermore, SIAM provides flexibility for both custom and homogeneous (generic) IMC architectures, allowing a higher degree of freedom for the user. Table 1 shows the comparison between different popular IMC simulators. GenieX [6] and RxNN [15] target monolithic IMC architectures. The performance of the computing fabric is obtained through SPICE-based models. Neither of these simulators estimate the performance of the communication fabric. NeuroSim [29] and MNSIM [44] utilize an H-tree based point-to-point (P2P) network and mesh-NoC, respectively as communication fabric. Both these simulators target monolithic IMC architectures. In contrast to all these simulators, SIAM provides the first chiplet-based IMC simulator that supports circuit, interconnect (NoC and point-to-point), DRAM cost estimation, and NoP interconnect evaluation.

3 CHIPLET-BASED IMC ARCHITECTURE

Overview: This section presents the underlying chiplet-based IMC architectures supported by SIAM, for *homogeneous (generic)* and *custom designs*. In a homogeneous architecture, the number of chiplets is fixed and is determined by the user. A custom architecture consists of the required

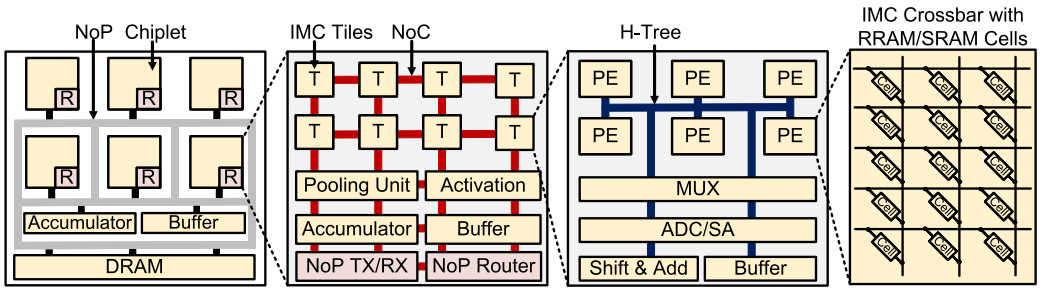

Fig. 2. Chiplet-based IMC architecture utilized within SIAM. The architecture consists of IMC chiplets, global accumulator, buffer, and DRAM connected using an NoP. SIAM supports both SRAM- and RRAM-based IMC architectures. An NoP is applied for inter-chiplet communication and an NoC is utilized within the chiplet for intra-chiplet communication.

number of chiplets to map the DNN under consideration. In both cases, the chiplet structure has a fixed number of IMC crossbar arrays inside (user-defined).

Figure 2 shows a homogeneous chiplet-based IMC architecture utilized by SIAM. The entire architecture consists of an array of chiplets that include IMC compute units, a global accumulator, a global buffer, and a DRAM chiplet. The chiplets are connected using an NoP fabric. The global accumulator and buffers are used to perform the accumulation across chiplets, and the DRAM chiplet is used to store the pre-trained DNN weights. In this work, we assume that all weights are transferred to the IMC chiplets from the DRAM chiplet before performing the DNN inference, consistent with prior works [14, 20, 34].

Intra-Chiplet IMC Architecture: Each IMC chiplet consists of an array of IMC tiles connected using an NoC. The IMC tiles consist of an array of processing elements (PEs) or crossbar arrays. SIAM currently supports both RRAM- and SRAM-based IMC crossbar architectures. The IMC crossbars utilize analog computation to perform the multiply-and-accumulate (MAC) operation. Each IMC crossbar has associated peripheral circuitry (e.g. column multiplexers, analog-to-digital converter (ADC), shift and add circuits, etc.). A column multiplexer is used to share a flash ADC or sense amplifier (SA) across multiple columns of the IMC crossbar. The ADC converts the analog output from the IMC crossbar to the digital domain. Next, the ADC output is accumulated based on the bit significance using shifter and adder circuits to extract the computed MAC output. Finally, the overall result is generated by accumulating the outputs from each IMC crossbar across the entire input. Note that our architecture does not use a digital-to-analog converter (DAC), and it instead employs sequential bit-serial computing for multi-bit inputs. Furthermore, each chiplet consists of a pooling and activation unit. The pooling unit supports both max and average pooling operations, while the activation unit supports ReLU and sigmoid functions.

Interconnect: The IMC chiplets are connected at the tile-level (within chiplet) using an NoC. A point-to-point (P2P) interconnect, such as H-Tree, is used for communication at the PE-level. Each tile within the chiplet has a five-port router that performs the data scheduling and X–Y routing through the NoC. The NoC can be configured for multiple flit width and operating frequencies by the user. The array of chiplets are interconnected using an NoP that utilizes the interposer for routing. A passive interposer is implemented within SIAM where the interposer does not contain any active elements like repeaters. Each chiplet consists of a dedicated NoP transmitter and receiver (TX/RX) circuit and an NoP router. The router performs the packet scheduling and utilizes a dedicated port to transmit data to the TX/RX circuit. The custom TX/RX circuit can be configured for a given signaling technique to achieve data transfer across the NoP [22, 40]. The architecture

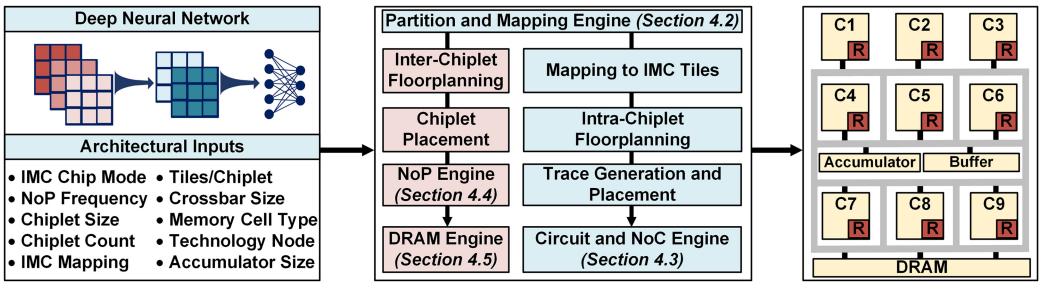

Fig. 3. An overview of the proposed chiplet-based IMC benchmarking simulator, SIAM. SIAM incorporates device, circuits, architecture, NoC [17], NoP, and DRAM access model [19] under a single roof for system-level analysis of chiplet-based IMC architectures.

Table 2. Definition of the User Inputs to SIAM

User Input	Description	User Input	Description
DNN Algorithm		**Device and Technology**	
Network Structure	DNN network structure information	Tech Node	Technology node for fabrication
Data Precision	Weights and activation precision	Memory Cell	RRAM or SRAM
Sparsity	DNN layer-wise sparsity	Bits/Cell	Number of levels in RRAM
Intra-Chiplet Architecture		**Inter-Chiplet Architecture**	
Crossbar Size	IMC crossbar array size	Chip Mode	Monolithic or chiplet-based IMC architecture
Buffer Type	SRAM or Register File	Chiplet Structure	Homogeneous or custom chiplet structure
ADC Resolution	Bit-precision of flash ADC	Chiplet Size	Number of IMC tiles within each chiplet
Read-out Method	Sequential or Parallel	Total Chiplet Count	Fixed count or DNN specific custom count
NoC Topology	Mesh or Tree	Global Accumulator Size	Size of global accumulator
NoC Width	Number of channels in the NoC	NoP Frequency	Frequency of the NoP driver and interconnect
Frequency	Frequency of operation	NoP Channel Width	Number of parallel links for TX and RX

also includes a clocking circuit (e.g.: LC-PLL [30]) for the TX/RX circuit. The NoP interconnect properties such as wire resistance, capacitance, and inductance are carefully modeled by utilizing the PTM models [37] following prior works [22, 40]. Section 4.4 details the modeling of both the NoP interconnect and the driver.

4 SIAM SIMULATOR

4.1 Overview

SIAM provides a unified framework for performance benchmarking of chiplet-based IMC architectures, as shown in Figure 3. SIAM operates on user inputs to generate the chiplet-based IMC architecture and to benchmark the corresponding hardware performance. The hardware performance metrics include area, energy, latency, energy efficiency, power, leakage energy, and IMC utilization. The overall simulator is developed using Python and C++ programming languages. A top-level Python wrapper is built to combine the different components within the simulator. Furthermore, SIAM interfaces with popular deep learning frameworks such as PyTorch and Tensorflow. Thus, SIAM supports multiple network structures in current literature (as shown in Section 6) and can be used for exploring NAS techniques. Table 2 shows the user inputs and associated descriptions of the SIAM benchmarking tool.

SIAM consists of four engines:

- Partition and mapping engine (Python)
- NoP engine (Python and C++)
- Circuit and NoC engine (C++)
- DRAM engine (Python and C++)

Each engine functions independently on a subset of the user inputs, while communicating with each other using the top-level Python wrapper. To further understand the framework, we detail

ALGORITHM 1: Partition and Mapping of DNN layers

Input: DNN structure, chiplet count (C), chiplet size (S), the number of DNN layers (N_l)
Output: Layer-wise chiplet partition ($\mathcal{L} \rightarrow \mathcal{P}$) and layer to chiplet mapping ($\mathcal{L} \rightarrow C$)
for $i = 1 : N_l$ **do**
    $N^{Chiplet} = 0,\ N_i^{Chiplet} = 0,\ N_i^{Total} = 0$ /* **Initialize variables** */
    /* **Layer-wise Mapping** ($\mathcal{L} \rightarrow C$) */
    Calculate number of rows of IMC crossbars (N_i^r) to map layer i (Equation 1)
    Calculate number of columns of IMC crossbars (N_i^c) to map layer i (Equation 1)
    $N_i^{Total} = N_i^r \times N_i^c$ /* **Total number of IMC crossbars for layer** i */
    /* **Layer-wise Partitioning** ($\mathcal{L} \rightarrow \mathcal{P}$) */
    $N_i^{Chiplet} = \left\lceil \frac{N_i^{Total}}{S} \right\rceil$ /* **Calculate the number of chiplets for layer** i */
    $N^{Chiplet} += N_i^{Chiplet}$ /* **Increment total chiplets in the architecture** */
    if *Homogeneous Mapping* **then**
        if $N^{Chiplet} > C$ **then**
            exit() /* **Error: Exceeded the maximum number of chiplets** */
        end
    end
    /* **Partition and Mapping completed for layer** i */
end

the simulation flow used for SIAM in Figure 3. *First*, SIAM takes the user inputs and performs the layer partition and mapping onto the chiplets and IMC crossbars using the partition and mapping engine. The outputs include the structure of IMC architecture, the number of chiplets and IMC tiles required per layer, utilization of the IMC architecture, intra-chiplet and inter-chiplet data movement volume, and the number of global accumulator accesses. Next, the intra-chiplet and global circuit performance are evaluated using the circuit and NoC engine. The engine provides the hardware performance metrics such as area, energy, and latency for the intra-chiplet and global circuit operations across all chiplets. Simultaneously, the NoP engine evaluates the cost of the interconnect, router, and driver associated with the chiplet-to-chiplet data movement. Finally, the DRAM engine determines the cost of the memory accesses and provides the energy and latency performance metrics. All engines except the partition and mapping engine work simultaneously, thus reducing the total simulation time. We note that SIAM also supports benchmarking of conventional monolithic IMC architectures. In the following sections, we detail the four engines that represent the core functionality within SIAM.

4.2 Partition and Mapping Engine

Algorithm 1 describes the step-by-step operation of the partition and mapping engine. The engine performs the partition of DNN layers to the IMC chiplets and the corresponding mapping to the IMC crossbar arrays. The partition and mapping is performed layer-wise for the entire DNN. The engine utilizes user inputs such as the DNN structure, DNN weight precision, IMC chiplet mapping scheme, size of the IMC chiplet, and the IMC crossbar size, among others.

We first discuss the IMC mapping scheme utilized in SIAM. For a given layer i, let the weight matrix be W_i represented by $\mathrm{Kx}_i \times \mathrm{Ky}_i \times \mathrm{Nif}_i \times \mathrm{Nof}_i$, where Kx and Ky represent the kernel size, Nif the number of input features, and Nof the number of output features. We adopt the following

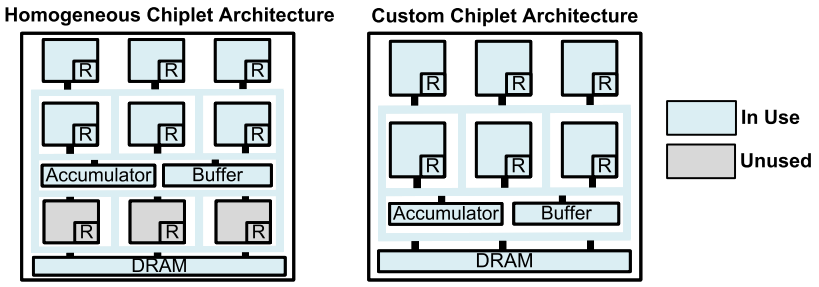

Fig. 4. Representative figure showing the two generated chiplet-based IMC architectures for the same DNN, homogeneous (left) and custom (right), from the supported partition schemes in SIAM. Homogeneous architecture is generic, while custom architecture is DNN specific. R refers to the NoP router.

mapping scheme, similar to that in [20, 34]:

$$N_i^r = \left\lceil \frac{Kx_i \times Ky_i \times Nif_i}{(PE_x)} \right\rceil; \qquad N_i^c = \left\lceil \frac{Nof_i \times N_{bits}}{(PE_y)} \right\rceil \tag{1}$$

In the above equation, N_i^r and N_i^c are the required number of rows and columns of IMC crossbars needed to map the layer i of the DNN. N_{bits}, PE_x and PE_y represent the DNN weight precision, the number of rows and columns in the IMC crossbar array, respectively. The product of N_i^r and N_i^c is the total number of required IMC crossbar arrays N_i^{Total} to map layer i of the DNN (line 7 of Algorithm 1).

SIAM can generate (a) *homogeneous* and (b) *custom* chiplet-based IMC architectures using two types of chiplet partitions. Figure 4 shows the two generated architectures based on the homogeneous and custom chiplet partitioning. The partition and mapping engine assumes that DNN layers cannot be partitioned across multiple chiplets, and a single chiplet can support multiple layers to achieve high chiplet utilization (shown in Section 6). Since each layer of the DNN contains a large number of multi-bit weights, multiple chiplets consisting of IMC crossbar arrays are required to map the whole layer. If one layer is distributed across chiplets, it increases the overhead in terms of the control logic for routing the inputs to the respective chiplets, an increased volume of inter-chiplet data communication, and a higher chiplet-to-chiplet communication energy and latency. During the partition of layers across multiple chiplets, the engine divides the layer uniformly across the chiplets, thus avoiding the workload imbalance issue. Based on the total number of required IMC crossbar arrays, N_i^{Total}, the engine determines the number of chiplets necessary to map the layer i of the DNN as: $N_i^{Chiplet} = \lceil \frac{N_i^{Total}}{S} \rceil$, where S denotes the total number of IMC crossbar arrays within a chiplet (size of the chiplet).

Next, the total number of chiplets in the architecture ($N^{Chiplet}$) is determined (line 9 of Algorithm 1). In the *homogeneous chiplet partition* scheme, a fixed number of chiplets (user input) is used to map the DNN. Hence, the engine compares the total number of chiplets in the architecture ($N^{Chiplet}$) with the maximum available chiplets (C). If greater, then the engine throws an error and requests for an increase in the number of available chiplets in the architecture. If lesser, the engine continues the partition and mapping for the subsequent layers in the DNN.

In the *custom partition scheme*, the architecture consists of the required number of chiplets to map the DNN. Hence, there is no maximum limit in the number of available chiplets within the architecture. Such a design results in a fully-custom architecture specific to the DNN under consideration. Each chiplet has the same structure with a fixed number of IMC tiles, where each tile consists of IMC crossbar arrays and associated peripheral circuitry. Thus, SIAM

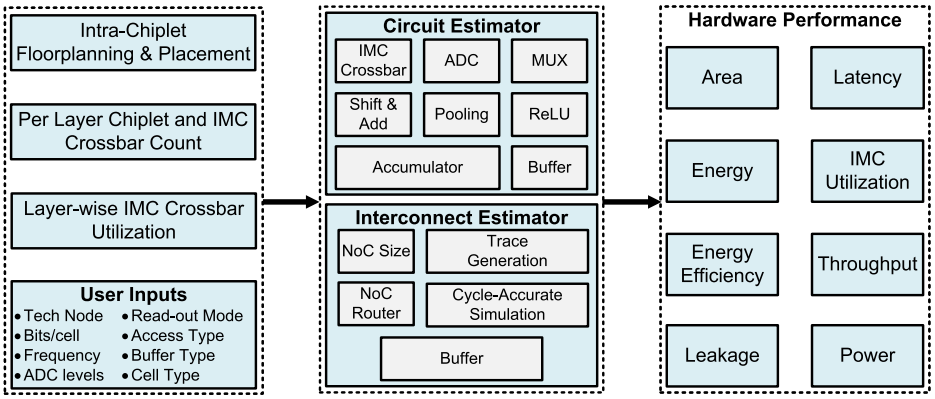

Fig. 5. Block diagram of the Circuit and NoC engine within SIAM. The engine utilizes a separate circuit and NoC simulators that perform the overall hardware performance estimation.

provides a platform to perform comparison between homogeneous (generic) and custom-designed chiplet-based IMC architectures.

After partitioning and mapping layers onto the IMC chiplets, the engine determines the total volume of data communicated within the chiplet and across chiplets. Simultaneously, when a layer is partitioned across chiplets, the global accumulator is used to generate the overall layer output. The engine determines the number of additions performed by the global accumulator and the number of global buffer accesses. Overall, the engine provides the layer partition across chiplets, the number of required chiplets and IMC crossbars, IMC crossbar utilization, volume of intra- and inter-chiplet data movement, and the number of the global accumulator and buffer accesses. The other engines (circuit, NoC and NoP) then utilize these outputs to evaluate the hardware performance of the chiplet-based IMC architecture.

4.3 Circuit and NoC Engine

After completing the partition and mapping of the DNN, SIAM performs the inter- and intra-chiplet floorplanning and placement, thus determining the entire chiplet-based IMC architecture. Thereafter, the circuit and NoC engine estimates the hardware performance. Figure 5 shows the block diagram of the circuit and NoC engine. The engine employs a model-based estimator for the circuit part, and a trace-based estimator for the interconnect part.

4.3.1 Circuit Estimator. The circuit estimator evaluates the overall hardware performance of each chiplet, global accumulator, and global buffer in the overall architecture. The inputs to the engine include the intra- and inter-chiplet placement, per layer chiplet and IMC crossbar count, layer-wise IMC utilization, technology node, frequency of operation, IMC cell type, the number of bits per cell, read-out mode (row-by-row or parallel), and ADC precision, among others. The intra-chiplet circuits include the IMC crossbar array and associated peripheral circuits, buffer, accumulator, activation unit, and the pooling unit. The peripheral circuits include the ADC, multiplexer circuit, shift and add circuit, and decoders. We calibrate the circuit estimator with NeuroSim [29].

The circuit estimator evaluates the performance of the entire chiplet-based IMC architecture in a layer-wise manner. Each chiplet performs the computations of a subset of layers in the DNN. For a given DNN layer i, the chiplet count per layer, the IMC crossbar count per layer, and the IMC utilization values are taken from the partition and mapping engine. Area, energy, and latency are estimated in a bottom-up manner, i.e., the estimation starts from the device level and moves

ALGORITHM 2: NoC (or NoP) Trace Generation

Input: Number of tiles for each layer (for each chiplet in case of NoP) ($|\mathcal{T}|$), Number of input activations for each layer(A), Number of chiplets (C), Layer to chiplet mapping ($\mathcal{L} \rightarrow C$), Quantization bit ($Q$), Bus width ($W$)
Output: Trace file for each chiplet (tr^c)
for $c = 1 : |C|$ **do**
  Find index of the first layer (L_c^S) and the last layer (L_c^E) in the chiplet from $\mathcal{L} \rightarrow C$
  for $l = L_c^S : L_c^E$ **do**
    $k = 0$ /* **Initialize timestamp** */
    Find index of first source tile (T_l^S) and last source tile (T_l^E)
    Find index of first destination tile (T_{l+}^S) and last destination tile (T_{l+}^E)
    $N_p = \lceil \frac{A(l)Q}{W} \rceil$ /* **Number of packets** */
    for $n = 1 : N_p$ **do**
      for $s = T_l^S : T_l^E$ **do**
        for $d = T_{l+}^S : T_{l+}^E$ **do**
          $tr^c \leftarrow [tr^c; [s, d, k]]$
          $k \leftarrow k + 1$ /* **Increment timestamp** */
        end
        $k \leftarrow k + 1$ /* **Increment timestamp** */
      end
    end
  end
end

up to the circuit level and, finally, the architecture level. Based on user inputs such as technology node, IMC cell type, IMC crossbar size, IMC utilization, ADC precision, and read-out mode, the estimator evaluates the cost of a single crossbar and associated peripheral circuits. The estimator repeats the process for all IMC crossbars within the chiplet for the given layer i in the DNN. Next, the estimator evaluates the buffer cost, shift and adder circuitry, and the accumulator within the chiplet. After that, the pooling and activation units are evaluated to obtain the total area, energy, and latency of the IMC chiplet. At the chiplet-level, the global accumulator and global buffer accumulate the partial sum of a layer across chiplets. The circuit estimator utilizes the number of additions performed, the data volume from each chiplet, and the accumulator size (user input) to determine the area, energy, and latency of the global accumulator and buffer. Finally, based on the number of chiplets required for layer i of the DNN, the circuit estimator repeats the estimation for all chiplets to determine the overall hardware performance.

4.3.2 NoC Estimator. Communication plays a crucial role in the hardware performance of DNN accelerators [23]. A detailed description of communication-centric DNN accelerators can be found in [28]. Each layer within the DNN sends a significant amount of data to other layers. Authors in [20] show that communication alone incurs up to 90% of the total inference latency for DNNs. Therefore, designing an efficient communication protocol for DNNs is of supreme importance. Hence, we carefully incorporate the cost of communication between multiple layers within a chiplet. We consider an NoC for intra-chiplet communication since NoC is the standard interconnect fabric used in the SoC-domain [16, 24]. We customize a cycle-accurate NoC simulator, BookSim [17], to evaluate the NoC performance. First, a trace file is generated for each chiplet following Algorithm 2. The algorithm takes the number of tiles for each layer, the number of input

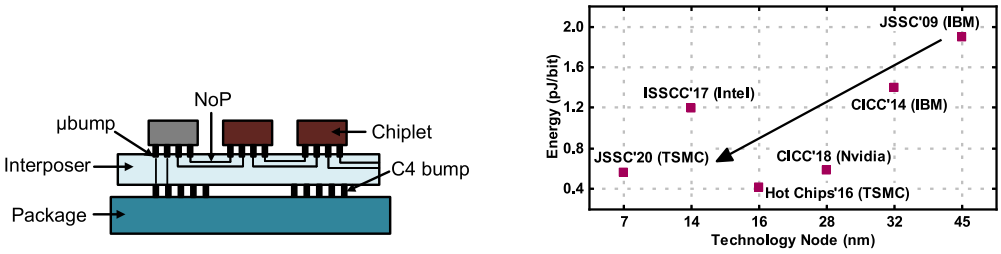

Fig. 6. (Left) Cross-sectional image of the NoP interconnect. The NoP is routed within the interposer connecting different chiplets across the architecture. μbumps connect the chiplets to the interposer, (Right) Energy per bit for different NoP driver circuit and signaling techniques proposed in prior works.

activations for each layer, number of chiplets, layer to chiplet mapping, quantization bit-precision, and bus width. From these inputs, we find the indices of the first and the last layer of each chiplet. Next, for each layer in each chiplet, we find the source and destination tile information as shown in lines 7–8 of Algorithm 2. The number of packets for each source-destination pair is then calculated. After that we iterate over the number of packets, the number of source tiles, and the number of destination tiles to obtain a trace in the form of a tuple consisting of the source tile ID, destination tile ID, and the timestamp. The timestamp variable is reset to zero after generating trace for each pair of layers. Then, the trace file is simulated using BookSim to obtain the area, energy, and latency for on-chip communication within each chiplet.

4.4 NoP Engine

The NoP connects different chiplets through a silicon interposer or organic substrate. It performs the on-chip data movement using special signaling techniques and driver circuits, as shown in [22, 30]. Figure 6 (left) shows the cross-sectional image of a 2.5D integration with chiplets and an interposer. Modeling the NoP performance has many challenges due to the complex interconnect structure, specialized driver architectures, and the corresponding signaling techniques.

To this end, our NoP engine models each component in the NoP for accurate performance estimation. Figure 6 (right) shows various NoP implementations with the corresponding energy-per-bit (E_{bit}) proposed in prior works. There are two main components of the NoP performance evaluation: 1) NoP latency estimation and 2) NoP area and power estimation.

NoP latency estimation: The engine utilizes a cycle-accurate simulator to perform the interconnect evaluation. First, based on the chiplet-to-chiplet data volume generated by the partition and mapping engine, the NoP engine utilizes Algorithm 2 (same as the algorithm for NoC) to generate the trace for the NoP. These traces are simulated using a cycle-accurate simulator or the NoP estimator (a customized version of BookSim to incorporate a trace-based simulation) to obtain the latency of the NoP interconnect.

NoP area and power estimation: To estimate the area and power consumption of the NoP, we first obtain the interconnect parameters for the NoP, which include wire length, pitch, width, and stack-up. We use these parameters to determine the interconnect capacitance and resistance using the PTM interconnect models [37]. Next, based on the capacitance and resistance, the timing parameters for the interconnect are generated and compared with the target bandwidth. If the timing parameters do not satisfy the bandwidth, the NoP engine chooses the maximum allowable bandwidth.

Next, the engine evaluates the NoP transmitter/receiver (TX/RX) circuits, including the clocking circuitry. The engine utilizes E_{bit}, number of TX/RX channels, bandwidth, chiplet-to-chiplet data

ALGORITHM 3: Computation of NoP driver energy

1 **Input:** DNN structure, Chiplet count (C), Number of activations per layer (A), NoP bus width (W), Quantization bit (Q_{bit}), Energy per bit (E_{bit})
2 **Output:** Energy for NoP driver (E_D)
3 **Initialize:** $E_D \leftarrow 0$
4 **for** $c = 1 : |C|$ **do**
5 Find index of source layer (l)
6 Find index of destination layer ($l + 1$)
  `/* Number of packets between two consecutive chiplets */`
7 $N_p = \lceil \frac{A(l)Q}{W} \rceil$
8 $N_{bits} = N_p \times Q_{bit}$
9 $E_D = E_D + N_{bits} \times E_{bit}$
10 **end**

volume, and operation frequency to generate the energy and latency cost of the TX/RX circuits. Algorithm 3 details the energy calculation for the NoP driver. We compute the total number of bits between chiplets. Furthermore, we obtain the energy per bit (E_{bit}) from prior works, as shown in Figure 6(right). We multiply the number of bits and energy per bit to obtain the total energy for TX/RX channel, as shown in line 9 of Algorithm 3. Next, the TX/RX circuit area from prior implementations (Figure 6) is utilized to obtain the NoP driver area cost. Finally, the NoP engine combines the performance metrics for the interconnect and the driver to generate the overall NoP performance. We summarize the functional flow of the NoP engine:

- NoP trace generation based on the inter-chiplet layer partition, chiplet placement, and inter-chiplet data transfer volume.
- NoP interconnect evaluation using a cycle-accurate simulator to generate area, energy, and latency metrics.
- NoP TX/RX driver and router modeling based on real measurements. Finally, the NoP engine combines the interconnect and NoP driver metrics to generate the overall NoP performance.

4.5 DRAM Engine

The chiplet-based IMC architecture consists of a DRAM chiplet that acts as the external memory for the IMC chiplets. The DRAM engine performs the external memory access estimation for the chiplet-based IMC architecture. In this work, we assume that the DRAM only transfers the entire set of weights to the chiplet one time before the inference task is performed. Hence, it remains constant for a given DNN across different architectural configurations and inference runs.

The engine consists of a DRAM request generator, RAMULATOR [19] for estimating the latency for the DRAM transactions, and VAMPIRE [8] to estimate the DRAM transaction power. First, the choice of DRAM is determined based on the user input. Currently, SIAM supports both DDR3 and DDR4. For DDR3 and DDR4, we incorporate the DRAM models detailed in [26, 27]. Next, for a given DNN model, the model size and data precision are determined from the user inputs. Furthermore, the DRAM engine generates the required traces and memory requests with time stamps. The requests include the location within the DRAM memory and the operation.

SIAM utilizes a customized version of the cycle-accurate simulator RAMULATOR [19] and the model-based power analysis tool VAMPIRE [8]. The customization includes the addition of support for larger DNNs, different data precision, and the addition of custom DDR3/DDR4 models. Furthermore, to reduce the simulation time for large DNNs such as VGG-16 (138M parameters),

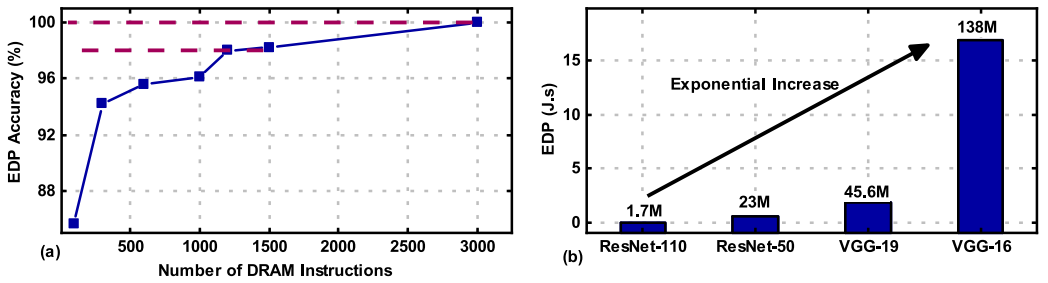

Fig. 7. (a) The accuracy of EDP prediction for different numbers of instructions processed to represent 3,000 DRAM instructions. Reduction in the number of instructions to half results in less than 2% EDP accuracy degradation for half the simulation time, and (b) EDP of DRAM transactions (DDR4) for different DNNs. There is an exponential increase in DRAM cost with an increase in DNN model size.

the DRAM engine breaks down the total number of instructions into smaller sets. The engine then performs the estimation for one set of instructions and multiplies it by the total number of sets required to represent all the weights in the DNN. To calibrate the method, we perform an experiment for 3,000 instructions broken down into a number of smaller sets of instructions. Figure 7(a) shows the corresponding energy-delay-product (EDP) accuracy for different sizes of instruction sets. A reduction in 50% of DRAM instructions to the engine results in less than 2% EDP accuracy degradation than that at 100% instructions. Furthermore, the reduced number of instructions allows for reduced simulation time for the DRAM engine. We establish that, through this method, the DRAM engine performs fast and accurate estimation of external memory access for the entire range of DNNs. Figure 7(b) shows the overall EDP for different networks across different datasets for DDR4. There is an exponential increase in EDP with the increase in the model size of the DNN.

To summarize, the following are the key steps in the execution of the DRAM engine:

- Generate DRAM requests based on the DNN model size and data precision.
- Calculate DRAM transaction latency cost using a customized version of RAMULATOR, and calculate the power consumption using a customized version of VAMPIRE.
- Combine the outputs to generate the overall DRAM access cost.

5 SIAM DATAFLOW

This section presents the default dataflow in the generated chiplet-based IMC architecture. Figure 8 shows an example of the computation dataflow within the SIAM architecture. Before performing the inference task, the weights are retrieved from the DRAM and mapped to the IMC chiplets based on the output from the partition and mapping engine (detailed in Section 4.2), as shown in Figure 8(a).

Two cases can arise during the partitioning: first, no layer is distributed across two or more chiplets; second, a layer is distributed across two or more chiplets. The two cases result in two different scenarios within the execution dataflow. Consider that layer N of the DNN is mapped onto the first chiplet in the architecture, as shown in Figure 8(a). During the computation, the entire layer is consumed within one chiplet, producing the computed output activations from layer N. Both the global accumulator and buffer are not utilized in the process and are turned off. After the computation, the output activations are transferred to the chiplets that implement layer N+1. For layer N+1, let's assume that two chiplets are required to map the weights. Hence, the NoP transfers the output activation from layer N to both chiplets housing layer N+1, as shown in Figure 8(a). Figure 8(b) shows the computation flow for layer N+1. Both chiplets perform the computation in a

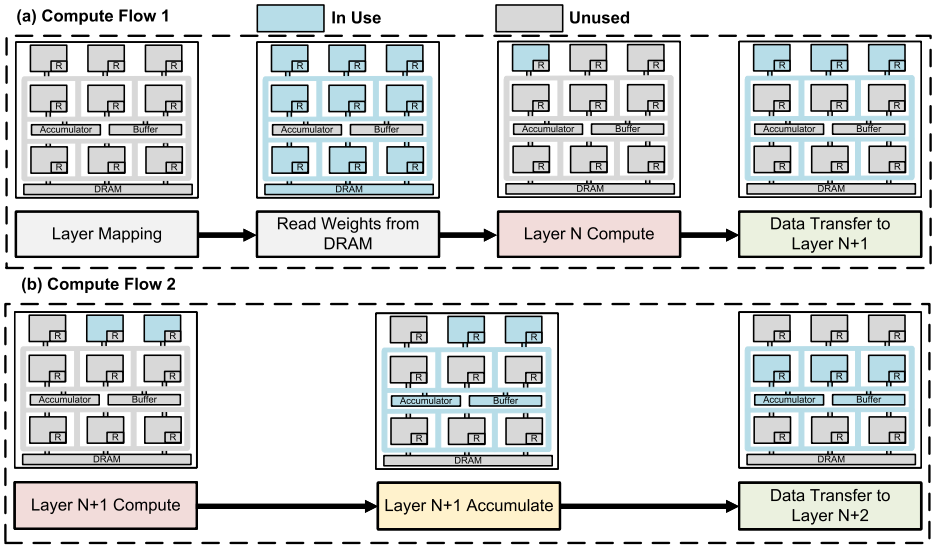

Fig. 8. Computation dataflow within the chiplet-based IMC architecture in SIAM. Two cases arise, (a) no layer is partitioned across two or more chiplets, and (b) a layer is partitioned across two or more chiplets.

ALGORITHM 4: Dataflow for SIAM IMC Chiplet Architecture

Input: Weights, input features, and total number of layers of the DNN
Output: Execution flow of the DNN on SIAM chiplet-based IMC architecture
while $i <$ *Total number of layers* **do**
    `/* Partitioning and mapping of the DNN */`
    Calculate number of chiplets required for i^{th} layer
    Perform computation for i^{th} layer
    `/* Check if layer is distributed across chiplets */`
    if *Number of chiplets > 1* **then**
        Data transfer to accumulator
        Partial sum accumulation for i^{th} layer
    end
    Data transfer to chiplets of $(i+1)^{\text{th}}$ layer
    $i \longleftarrow i+1$
end

parallel manner. The mapping ensures that the same number of weights are mapped to each chiplet, thus avoiding the workload imbalance issue. After completion of the computation, the generated partial sums are accumulated using the global accumulator and buffer. Then, the accumulated outputs from layer N+1 are transferred to the chiplets housing weights of layer N+2. The process is repeated until all the layers are completed and the final output is obtained. Algorithm 4 details the algorithmic implementation of the dataflow utilized in the SIAM IMC chiplet architecture.

6 EXPERIMENTAL EVALUATION

We perform a wide range of experiments to demonstrate the effectiveness of the proposed SIAM simulator. These include detailed analysis of homogeneous and custom IMC chiplet architectures,

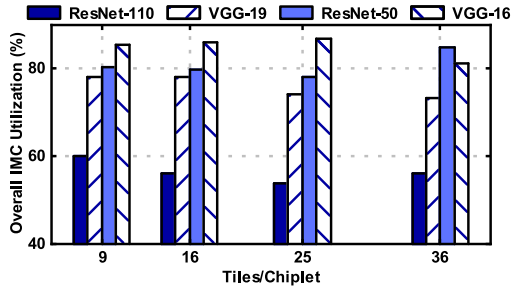

Fig. 9. IMC utilization for a custom RRAM-based chiplet IMC architecture across different DNNs and different chiplet configurations. The mapping strategy adopted within SIAM ensures high utilization across all DNNs.

comparison between monolithic and chiplet IMC architectures, calibration with real silicon data from SIMBA [35], comparison of the performance with GPUs, and evaluation of the SIAM's simulation time. We also illustrate the three characteristics of SIAM, flexibility, scalability, and simulation speed as shown below:

- *Flexibility and scalability:* Supporting different DNNs across datasets, two types of DNN partition to the IMC chiplets, and support for different IMC tile and chiplet configurations (Section 6.2, Section 6.3).
- *Simulation speed:* Fast design space exploration of SIAM is demonstrated for different DNNs (Section 6.6).

6.1 Experimental Setup

The DNNs that we evaluated include ResNet-110 (1.7M) on CIFAR-10, VGG-19 (45.6M) on CIFAR-100, ResNet-50 (23M) on ImageNet, and VGG-16 (138M) on ImageNet. We use 8-bit quantization for the weights and activations and a 32nm CMOS technology node for the hardware. We perform the experiments on an Intel Xeon CPU platform. The mapping of DNNs onto the IMC crossbars follows prior works [20, 34]. Unless specified otherwise, all the experiments are performed based on the assumptions detailed next. The chiplets are placed to achieve the least Manhattan distance. All results shown are for RRAM-based IMC architectures with the following parameters: one bit per RRAM cell, a R_{off}/R_{on} ratio of 100, 16 tiles per chiplet, IMC crossbar size of 128×128, ADC resolution of 4-bits with 8 columns multiplexed, operating frequency of 1GHz [14, 34], and a parallel read-out method. We note that, the experiments do not consider the non-ideal effects within the RRAM-based IMC architecture. The NoP parameters include a bandwidth of 250MHz, E_{bit} of 0.54pJ/bit [30], interconnect parameters such as width, thickness, and pitch from [30], NoP TX/RX area of 5,304 μm^2 [30], NoP clocking circuit area of 10,609 μm^2 [30], and 32 channels (channel width). We note that SIAM can support any NoP performance estimation as long as the NoP wiring parameters, bandwidth, channel width, TX/RX circuit area, clocking circuit area, and E_{bit} are provided by the user. Finally, the reported results do not include the RRAM write, and DRAM read energy and latency. Since we focus on DNN inference, the RRAM write and DRAM read operations are applied offline before performing the inference. They do not involve in the inference runs and remain constant across IMC chiplet architectural configurations.

6.2 Custom and Homogeneous Chiplet-based IMC Design

6.2.1 IMC Crossbar Utilization. Figure 9 shows the overall IMC crossbar utilization for a custom RRAM-based chiplet IMC architecture across different tiles per chiplet and DNNs. We see that

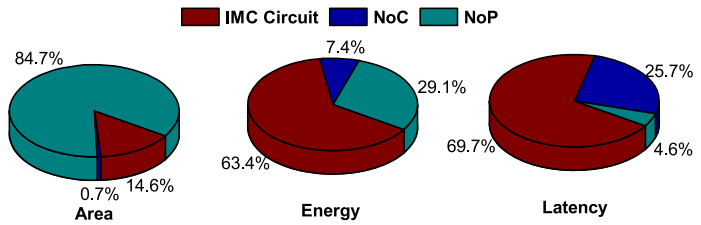

Fig. 10. Breakdown of the different components contributing to the overall area, energy, and latency performance metrics, for a custom design RRAM-based chiplet IMC architecture when mapping ResNet-110 for CIFAR-10 dataset.

SIAM consistently achieves high (>50%) IMC crossbar utilization. The high IMC utilization indicates that mapping within SIAM generates chiplet-based IMC architectures that are area-efficient. ResNet-110 has the lowest utilization due to the small network structure with fewer input and output features. At the same time, ResNet-50, VGG-19, and VGG-16 achieve >75% utilization across the entire chiplet-based IMC architecture. Hence, the partition and mapping engine within SIAM provides a flexible and efficient platform to generate chiplet-based IMC architectures with multiple configurations for design space exploration.

6.2.2 IMC Chiplet Performance Breakdown. We analyze the breakdown of different components for the area, energy, and latency metrics for the RRAM-based chiplet IMC architecture. Figure 10 shows the breakdown for the implementation of ResNet-110 on CIFAR-10. We divide each metric into three main components, IMC circuit, NoC, and NoP. The IMC circuit component consists of the IMC crossbar array and associated peripherals, buffers (global and within chiplet), accumulators (global and within chiplet), pooling unit, and the activation unit. At the same time, the NoP component consists of the NoP interconnect, NoP router, and the NoP driver and clocking circuit. Finally, the NoC component deals with the intra-chiplet interconnect and the NoC routers.

We first analyze the area metric. The NoP dominates the overall area with 84.7%, while the NoC contributes the least to the area. The NoP drivers are designed such that differential signaling is utilized to avoid common-mode noise along with a clocking circuit for every N lanes. This results in increased circuitry for the TX-RX driver pairs and associated clocking circuitry for the 32 NoP channels. For example, [35] utilizes one clocking lane per 4 data lanes. The NoP router area depends on the technology node of the chiplet and the number of ports (default ports is set to 5). Simultaneously, the NoP link area depends on the wire properties. The NoP wire width and length are designed to maintain signal integrity at the specified frequency of operation. The wire for NoP link requires shielding on both sides of the signal, thus resulting in an increased pitch [30]. This results in a significant increase in the wiring area. We note that the NoP wire has a 56x larger metal pitch than that for the wires within the chiplet. Furthermore, an increased NoP channel width is needed for higher performance at the cost of increased area. For energy and latency, the IMC circuit component dominates with 63.4% and 69.7% contributions, respectively. The NoP contributes the second highest to energy, while the NoC contributes the least. Simultaneously, NoC contributes the second highest to latency, while NoP contributes the least. Overall, the area is dominated by the NoP, and the energy and latency are dominated by the IMC circuit.

6.2.3 NoP and NoC Performance Trade-offs. We compare the EDAP for the NoC and NoP interconnect. Figure 11(a) shows the ratio of the EDAP of the NoP to that of NoC for ResNet-110 on CIFAR-10, for both homogeneous and custom RRAM-based chiplet IMC architectures. When there are fewer number of tiles per chiplet, more IMC chiplets are used to map the DNN to the

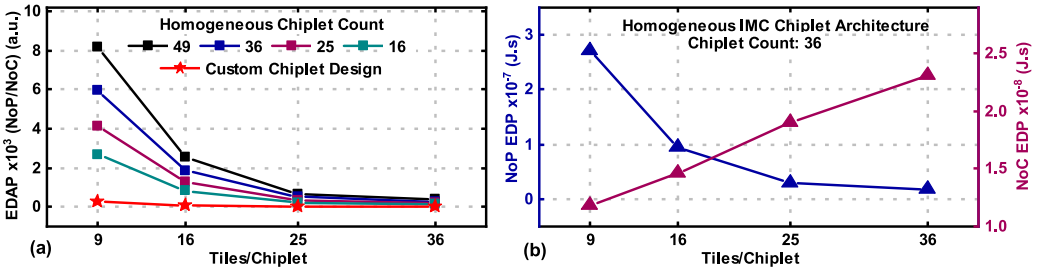

Fig. 11. NoP and NoC trade-off analysis for ResNet-110 on CIFAR-10 dataset. (a) Ratio of the energy-delay-area product (EDAP) of NoP to NoC for both homogeneous and custom chiplet-based IMC architectures. The increase in tiles per chiplet reduces the NoP/NoC EDAP, (b) NoP and NoC energy-delay product (EDP) for a 36 chiplet count configuration of homogeneous RRAM-based chiplet IMC architectures. An increased tiles per chiplet leads to higher NoC cost and lower NoP cost.

IMC crossbars, resulting in distributed computing. This results in an increase in the data transfer volume across chiplets and higher NoP EDAP compared to that of NoC. Furthermore, at higher chiplet counts, the NoP is much larger and results in increased area, thus increasing the EDAP. As we increase the number of tiles per chiplet, computations are more localized, leading to reduced volume of data transfer across chiplets. This reduces the ratio of NoP EDAP to NoC EDAP. The custom chiplet-based IMC architecture consists of the required number of chiplets to map the DNN under consideration. In addition, the custom chiplet architecture is designed specific to a DNN, resulting in a highly localized computing platform with a smaller NoP. Hence, the ratio of NoP EDAP to NoC EDAP is very small and is relatively insensitive to the change in tiles per chiplet.

To further understand the trade-off between NoC and NoP, we evaluate the energy-delay product (EDP) of NoC and NoP separately. Figure 11(b) shows the energy-delay product (EDP) for the NoP and NoC for a 36 chiplet count configuration of homogeneous chiplet IMC architecture. The x-axis shows the number of tiles in each chiplet. The EDP of NoP reduces with the increasing number of tiles in each chiplet. The reduced EDP of NoP is achieved due to the highly localized computing resulting in lesser inter chiplet communication data volume. At the same time, the NoC EDP increases with an increase in tiles per chiplet. The increased EDP is due to the larger NoC size (3x3 for 9 tiles per chiplet compared to 4x4 for 16 tiles per chiplet) and the increased intra-chiplet communication volume. Hence, a balance between the NoP and NoC cost is essential for optimal DNN inference performance with chiplet-based IMC architectures. We note that a similar trend is seen for other chiplet count configurations. From this experiment (ResNet-110 on CIFAR-10), we can conclude that the design with 16 tiles per chiplet provides a good balance in communication volume between NoC and NoP.

6.2.4 Overall Hardware EDAP and Area. Figure 12(a) shows the overall performance (EDAP) of the RRAM-based chiplet IMC architecture for ResNet-110 on CIFAR-10. For a homogeneous chiplet-based IMC architecture, the EDAP increases with higher chiplet counts (lower tiles per chiplet), resulting in higher chip area. Furthermore, with higher tiles per chiplet, the total energy contribution from the NoP reduces due to highly localized computing from the larger chiplet size and a lower NoP data volume. Overall, higher tiles per chiplet and lower chiplet count allow for a reduced EDAP in homogeneous RRAM-based chiplet IMC architectures. Simultaneously, the custom chiplet architecture has better performance than the homogeneous architecture due to the reduced NoP size and a customized architecture for the given DNN. A similar outcome arises for the custom design on increasing the tiles per chiplet, with the reduction in the EDAP.

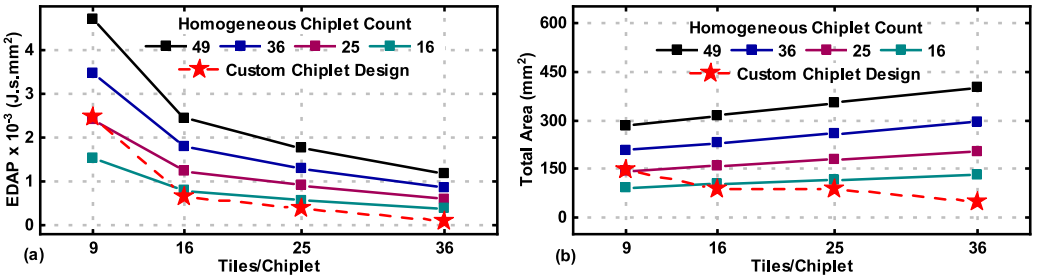

Fig. 12. Energy-delay-area product as the metric. (a) Overall EDAP and (b) total area for the homogeneous and custom RRAM-based chiplet IMC architecture when mapping ResNet-110 for CIFAR-10 dataset. The results indicate that a custom architecture outperforms a homogeneous architecture. The increased number of tiles per chiplet provides better performance at the cost of increased area for the homogeneous chiplet IMC architecture, while providing better performance and lower area for the custom chiplet IMC architecture.

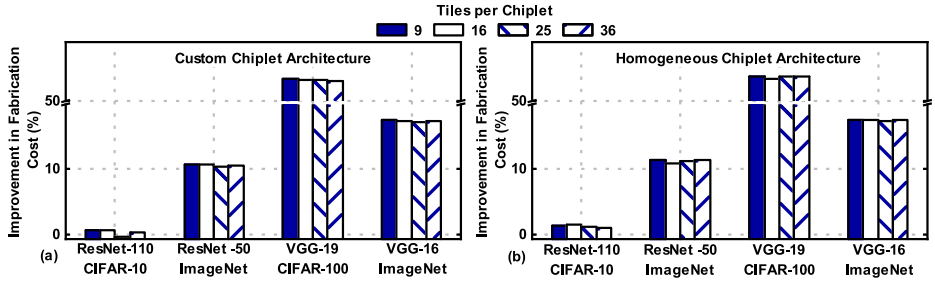

Fig. 13. Improvement in fabrication cost ($) for the RRAM-based chiplet IMC architecture, (a) custom and (b) homogeneous, as compared to a monolithic RRAM-based IMC architecture. Smaller DNNs like ResNet-110 have similar cost for both architectures, while larger DNNs such as VGG-19 have up to 60% improvement.

Figure 12(b) shows the overall area of the chiplet-based IMC architecture with different tiles per chiplet and different chiplet counts (homogeneous) and the custom chiplet architecture. The increase in the number of tiles per chiplet results in a higher area for the homogeneous chiplet architecture. The increase in area is due to the larger chiplet size while keeping the total chiplet count fixed. For example, the area for a 36 chiplet count and 16 tiles per chiplet architecture is larger than that of the 36 chiplet count and 9 tiles per chiplet architecture. Furthermore, the custom chiplet IMC architecture utilizes the required number of chiplet to map the whole DNN. Such an architecture benefits from increasing the tiles per chiplet since fewer chiplets are required to map the DNN. This results in lower NoP area and chiplet area (IMC circuit and NoC). Hence, with the increase in the number of tiles per chiplet the total area is reduced for the custom chiplet IMC architecture.

6.3 Comparison between Monolithic and Chiplet-based IMC Architectures

We perform a comparison between a custom monolithic RRAM-based architecture and both homogeneous and custom RRAM-based chiplet IMC architectures. Due to increased area, a monolithic IMC architecture suffers from increased defect ratio and lower yield. Consequently, a large monolithic IMC architecture experiences a very high fabrication cost, as shown in Figure 1(a). Figure 13 shows the improvement in fabrication cost of (a) custom and (b) homogeneous RRAM-based chiplet IMC architectures, compared to that of a monolithic RRAM-based IMC architecture.

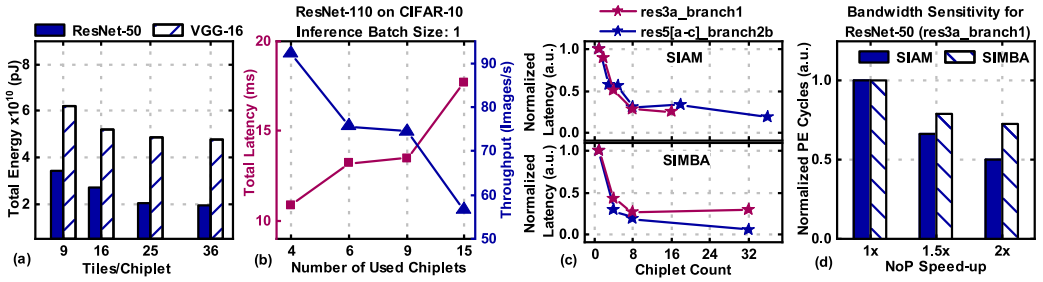

Fig. 14. (a) Total energy for DNN inference reduces with an increase in the number of tiles per chiplet (chiplet size), (b) Total inference latency and throughput for ResNet-110 on CIFAR-10. Due to the small network size, a lower number of chiplets provide better performance, (c) Normalized latency for two layers within ResNet-50 on ImageNet for SIAM RRAM-based chiplet IMC architecture and SIMBA [35]. The decreasing trend of latency with increasing chiplet count exhibited by SIAM is consistent with SIMBA, (d) Bandwidth sensitivity for a layer within ResNet-50. The decreasing trend in the PE cycles with increasing NoP speed-up similar to SIMBA.

We observe that the improvement is similar for different number of tiles per chiplet for a particular DNN. The increase in the number of tiles per chiplet results in a reduction in the total used chiplets, while keeping the same utilization across the used chiplets. Moreover, the improvement is similar for both custom and homogeneous chiplet architecture for a particular DNN. The improvement in fabrication cost is a strong function of the DNN structures. DNNs with fewer parameters exhibit less improvement. For example, ResNet-110 with 1.7M number of parameters shows up to 0.57% improvement in fabrication cost. At the same time, VGG-19 with 45.6M number of parameters show more than 50% improvement in the fabrication cost. The reduced fabrication cost is attributed to lower defect ratio and increased yield achieved through smaller chiplets connected together to form a large system. Hence, larger and branched DNNs benefit significantly from chiplet-based IMC architectures.

6.4 Calibration with SIMBA

This section presents the calibration results for the SIAM RRAM-based chiplet IMC architecture compared to published silicon data from SIMBA [35]. We note that, there is no prior work that reports real silicon data for chiplet-based IMC architectures. Therefore, we choose SIMBA, the work that has the most resemblance to that of SIAM. We utilize the NoP driver circuit and the signaling technique similar to those in SIMBA for our experiments. Furthermore, the NoP interconnect parameters utilized in SIAM are closed to that in SIMBA.

Total Energy: Figure 14(a) shows the total energy for inference across different number of tiles per chiplet for both ResNet-50 and VGG-16 on the ImageNet dataset. The increase in the number of tiles per chiplet results in a reduction in the total number of chiplets used to map the DNN and, in turn, a reduction in the total inference energy. The same trend is reported in SIMBA (ResNet-50).

Total Latency: We evaluate the effect of chiplet scaling or, in other words, the number of chiplets used to map a small DNN. Figure 14(b) shows the total inference latency and throughput for ResNet-110 on CIFAR-10 dataset. Since ResNet-110 is a small DNN, distributing the computation across more chiplets results in a sub-optimal configuration. A similar trend is shown in SIMBA for a small DNN, DriveNet [5].

Layer Sensitivity: We consider two representative layers in ResNet-50, res3a_branch1 and res5[a-c]_branch2b , same as that shown in SIMBA. We analyze the latency (normalized with the

Table 3. Simulation Time for SIAM

Network	Dataset	Model Size (M)	Simulation Time (Hours)
ResNet-110	CIFAR-10	1.7	0.2
VGG-19	CIFAR-100	45.6	0.36
ResNet-50	ImageNet	23	1.26
VGG-16	ImageNet	138	4.26

latency of the design consisting of only one chiplet) by varying the number of chiplets used to map the DNN layer. We note that the chiplet count to map the DNN is different from SIMBA due to the difference in the computation element in SIAM (IMC crossbars) and SIMBA (MAC arrays). The analysis shown in Figure 14(c) (top) reveals that there is a decreasing trend in latency with increasing number of chiplets. For res3a_branch1 the latency reduces initially with the increase in chiplet count and finally increases slightly (with chiplet count of 16). The increase in latency is due to a higher NoP latency from more distributed computation. res5[a-c]_branch2b shows a consistent decrease in the latency with the increase in chiplet count. These trends are consistent with that reported in SIMBA, as shown in Figure 14(c) (bottom).

PE cycles vs NoP speed-up: In this experiment, we vary NoP frequency and analyze the variation in PE cycles of res3a_branch1 layer in ResNet-50. We normalize it to the 1× case to be consistent with SIMBA. The SIAM chiplet IMC architecture shows decreasing PE latency with increasing NoP bandwidth, which conforms SIMBA, as shown in Figure 14(d).

In summary, these comparisons confirm that, during the scaling of chiplet parameters, such as the number of chiplets and their utilization, SIAM predicts similar trends as the measured results from real silicon.

6.5 Comparison with GPUs

We compare the performance of the chiplet-based IMC architecture generated using SIAM with state-of-the-art GPUs such as Nvidia V100 and T4. Unlike GPUs, SIAM generates architectures for energy-efficient inference using small batch sizes. All GPU hardware performance numbers have been adopted from those reported in [35]. We compare the performance of the architecture for inference with a batch size of one. For ResNet-50 on ImageNet dataset, the architecture generated using SIAM (36 tiles per chiplet) results in a total area of 273 mm^2 as compared to 525 mm^2 for T4 and 815 mm^2 for V100. The reduced area is attributed to the high compute density achieved using an IMC design and the support for a wide range of computations within the GPU. We also compare the energy-efficiency of SIAM IMC architecture to that of the GPUs for ResNet-50 on ImageNet. SIAM achieves 130× and 72× higher energy-efficiency as compared to V100 and T4 GPUs, respectively. The chiplet-based IMC architecture has higher performance since IMC architectures have all weights on chip, thus avoiding the external memory access. Furthermore, IMC utilizes analog domain computation within the crossbar arrays that are more energy-efficient than regular multiply-and-accumulate (MAC) units [34].

6.6 Simulation Time

Table 3 shows the simulation time for the proposed chiplet-based IMC simulator, SIAM, for different DNNs across different datasets. The simulation time is extracted by running SIAM on an Intel Xeon W-2133 CPU platform with 12 cores and 32GB RAM. The range of the simulation times varies from a couple of minutes for small DNNs to a few hours for large DNNs. For a fair analysis, we report the overall simulation time that includes the partitioning and mapping, circuit and NoC

simulation, NoP estimation, and DRAM access estimation. For example, ResNet-110 with 1.7M parameters for CIFAR-10 dataset takes 12 minutes (0.2 hours) for SIAM to perform the performance benchmarking. A large DNN such as VGG-16 with 138M parameters on the ImageNet dataset takes 4.26 hours for SIAM to perform the benchmarking.

Finally, we perform a comparison between SIAM and NeuroSim [29] in terms of simulation for benchmarking a RRAM-based monolithic IMC architecture. We note that we choose a monolithic IMC architecture as no other simulator supports chiplet-based IMC architecture benchmarking. We perform the comparison for four networks namely, ResNet-110 (1.7M), VGG-19 (45.6M), ResNet-50 (23M), and VGG-16 (138M). For ResNet-110 SIAM requires 60s while NeuorSim takes 30s while for VGG-19, SIAM takes 86s and NeuroSim takes 46s. Furthermore, for ResNet-50 SIAM takes 818s while NeuroSim takes 276s and for VGG-16 SIAM takes 3110s while NeuroSim takes 1110s. We note that the simulation times for SIAM are in the same range as that of NeuroSim rather than being orders of magnitude higher. The increased simulation times are due to the additional functionality that SIAM provides in NoC and DRAM performance estimation.

7 CONCLUSION AND DISCUSSION

This work presents SIAM, a novel performance benchmarking tool for chiplet-based IMC architectures. To the best of our knowledge, this will be the first benchmarking tool for design space exploration of chiplet-based IMC architectures. SIAM integrates device, circuits, architecture, NoC, NoP, and DRAM estimation into an end-to-end system. It supports two types of chiplet architectures, homogeneous and custom, generated using different partition schemes. In addition, SIAM supports different DNNs across different datasets and various IMC chiplet configurations. Through this study, we establish the scalability and flexibility features of SIAM. We demonstrate the speed of SIAM by evaluating the simulation time of SIAM for different DNNs and comparing it against state-of-the-art chiplet simulators like NeuroSim for monolithic IMC architectures. Next, we calibrate SIAM with respect to published silicon result, SIMBA, which confirms that the trends projected by SIAM match the measured results from real silicon. Finally, we compare the performance of the generated chiplet-based IMC architecture using SIAM to that of state-of-the-art GPUs. The SIAM chiplet architecture achieves 130× and 72× improvement in energy-efficiency compared to Nvidia V100 and T4 GPUs, respectively.

APPENDIX A

Estimating manufacturing cost of a chip: Let us consider a reference chip with area A_{ref} and N_{ref} chips per wafer. The cost of the reference chip (C_{ref}) is expressed as shown in Equation 2.

$$C_{ref} = \frac{C_{Total}}{\eta_{ref} N_{ref}} \tag{2}$$

where, η_{ref} is the yield of the wafer. Number of chips per wafer (N_{ref}) is expressed in 3 [1].

$$N_{ref} = D\pi \left(\frac{D}{4A_{ref}} - \frac{1}{\sqrt{2A_{ref}}} \right) \tag{3}$$

where D is the wafer diameter. The cost of a target (C_{target}) and normalized cost (C_{norm}) are:

$$C_{target} = \frac{C_{Total}}{\eta_{target} N_{target}} \qquad C_{norm} = \frac{C_{target}}{C_{ref}} = \frac{N_{ref}\eta_{ref}}{N_{target}\eta_{target}} \tag{4}$$

Assuming poisson defect model, $\eta = e^{-D_0 A}$, where D_0 is the defect density. Replacing the expression of η in expression for C_{norm}, we obtain:

$$C_{norm} = \frac{N_{ref} e^{-D_0 A_{ref}}}{N_{target} e^{-D_0 A_{target}}} = \frac{N_{ref}}{N_{target}} e^{-D_0 (A_{ref} - A_{target})} \tag{5}$$

Verification of chip cost estimation: To verify the chip cost estimation, we assume $A_{ref} = 296mm^2$, $D_0 = 0.012/mm^2$ and $D = 152.4mm$. The comparison reveals that the estimation is 98% accurate with respect to the real chip cost of commercial processors [39].

REFERENCES

[1] AnySilicon. 2011. Fabrication Cost. https://anysilicon.com/die-per-wafer-formula-free-calculators/. Accessed 29 Mar. 2021.

[2] Ali BanaGozar et al. 2019. CIM-SIM: Computation in memory SIMulator. In *Proceedings of the 22nd International Workshop on Software and Compilers for Embedded Systems*. 1–4.

[3] Noah Beck, Sean White, Milam Paraschou, and Samuel Naffziger. 2018. 'Zeppelin': An SoC for multichip architectures. In *2018 IEEE ISSCC*. IEEE, 40–42.

[4] Nathan Binkert et al. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

[5] Mariusz Bojarski et al. 2017. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911* (2017).

[6] Indranil Chakraborty, Mustafa Fayez Ali, Dong Eun Kim, Aayush Ankit, and Kaushik Roy. 2020. Geniex: A generalized approach to emulating non-ideality in memristive xbars using neural networks. In *2020 57th ACM/IEEE DAC*. IEEE.

[7] Marc Erett et al. 2018. A 126mW 56Gb/s NRZ wireline transceiver for synchronous short-reach applications in 16nm FinFET. In *2018 IEEE ISSCC*. IEEE, 274–276.

[8] Saugata Ghose et al. 2018. What your DRAM power models are not telling you: Lessons from a detailed experimental study. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2, 3 (2018), 1–41.

[9] David Greenhill et al. 2017. 3.3 A 14nm 1GHz FPGA with 2.5 D transceiver integration. In *2017 IEEE ISSCC*. IEEE.

[10] Kaiming He et al. 2016. Deep residual learning for image recognition. In *IEEE CVPR*. 770–778.

[11] Andrew Howard et al. 2019. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF ICCV*. 1314–1324.

[12] Gao Huang et al. 2017. Densely connected convolutional networks. In *IEEE CVPR*. 4700–4708.

[13] Ranggi Hwang, Taehun Kim, Youngeun Kwon, and Minsoo Rhu. 2020. Centaur: A chiplet-based, hybrid sparse-dense accelerator for personalized recommendations. In *2020 ACM/IEEE 47th Annual ISCA*. IEEE, 968–981.

[14] Mohsen Imani, Saransh Gupta, Yeseong Kim, and Tajana Rosing. 2019. Floatpim: In-memory acceleration of deep neural network training with high precision. In *Proceedings of the 46th Annual ISCA*. 802–815.

[15] Shubham Jain, Abhronil Sengupta, Kaushik Roy, and Anand Raghunathan. 2020. RxNN: A framework for evaluating deep neural networks on resistive crossbars. *IEEE TCAD* (2020).

[16] James Jeffers et al. 2016. *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*.

[17] Nan Jiang et al. 2013. A detailed and flexible cycle-accurate network-on-chip simulator. In *IEEE ISPASS*. 86–96.

[18] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H Loh. 2015. Enabling interposer-based disintegration of multi-core processors. In *2015 48th Annual IEEE/ACM MICRO*. IEEE, 546–558.

[19] Yoongu Kim, Weikun Yang, and Onur Mutlu. 2015. RAMULATOR: A fast and extensible DRAM simulator. *IEEE Computer Architecture Letters* 15, 1 (2015), 45–49.

[20] Gokul Krishnan et al. 2020. Interconnect-aware area and energy optimization for in-memory acceleration of DNNs. *IEEE Design & Test* 37, 6 (2020), 79–87.

[21] Gokul Krishnan et al. 2021. Interconnect-centric benchmarking of in-memory acceleration for DNNS. In *2021 China Semiconductor Technology International Conference (CSTIC)*. IEEE, 1–4.

[22] Mu-Shan Lin et al. 2020. A 7-nm 4-GHz Arm^1-core-based CoWoS^1 chiplet design for high-performance computing. *IEEE Journal of Solid-State Circuits* 55, 4 (2020), 956–966.

[23] Sumit K Mandal et al. 2020. A latency-optimized reconfigurable noc for in-memory acceleration of DNNs. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, 3 (2020), 362–375.

[24] Sumit K Mandal, Anish Krishnakumar, and Umit Y Ogras. 2021. Energy-efficient networks-on-chip architectures: Design and run-time optimization. *Network-on-Chip Security and Privacy* (2021), 55.

[25] Radu Marculescu et al. 2008. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 1 (2008), 3–21.

[26] MICRON. 2011. Datasheet for DDR3 Model. https://media-www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr3/2gb_ddr3l-rs.pdf?rev=f43686e89394458caff410138d9d2152. Accessed 29 Mar. 2021.

[27] MICRON. 2014. Datasheet for DDR4 Model. https://www.micron.com/-/media/client/global/documents/products/data-sheet/dram/ddr4/4gb_ddr4_dram_2e0d.pdf. Accessed 29 Mar. 2021.
[28] Seyed Morteza Nabavinejad et al. 2020. An overview of efficient interconnection networks for deep neural network accelerators. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, 3 (2020), 268–282.
[29] Xiaochen Peng, Shanshi Huang, Yandong Luo, Xiaoyu Sun, and Shimeng Yu. 2019. DNN+ neurosim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies. In *2019 IEEE IEDM*. IEEE, 32–5.
[30] John W Poulton et al. 2013. A 0.54 pJ/b 20Gb/s ground-referenced single-ended short-haul serial link in 28nm CMOS for advanced packaging applications. In *2013 IEEE ISSCC*. IEEE, 404–405.
[31] Yasir Mahmood Qureshi, William Andrew Simon, Marina Zapater, David Atienza, and Katzalin Olcoz. 2019. Gem5-X: A Gem5-based system level simulation framework to optimize many-core platforms. In *2019 SpringSim*. IEEE, 1–12.
[32] Enrico Russo, Maurizio Palesi, Salvatore Monteleone, Davide Patti, Giuseppe Ascia, and Vincenzo Catania. 2021. LAMBDA: An open framework for deep neural network accelerators simulation. In *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*.
[33] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. 2018. Scale-sim: Systolic CNN accelerator simulator. *arXiv preprint arXiv:1811.02883* (2018).
[34] Ali Shafiee et al. 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM/IEEE ISCA* (2016).
[35] Yakun Sophia Shao et al. 2019. SIMBA: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the 52nd Annual IEEE/ACM MICRO*. 14–27.
[36] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[37] Saurabh Sinha, Greg Yeric, Vikas Chandra, Brian Cline, and Yu Cao. 2012. Exploring sub-20nm FinFET design with predictive technology models. In *DAC 2012*. IEEE, 283–288.
[38] Linghao Song et al. 2017. Pipelayer: A pipelined reram-based accelerator for deep learning. In *IEEE HPCA*. 541–552.
[39] Simon M Tam et al. 2018. SkyLake-SP: A 14nm 28-core xeon® Processor. In *2018 IEEE ISSCC*. 34–36.
[40] Walker J Turner et al. 2018. Ground-referenced signaling for intra-chip and short-reach chip-to-chip interconnects. In *2018 IEEE CICC*. IEEE, 1–8.
[41] Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. 2019. Exploring randomly wired neural networks for image recognition. In *Proceedings of the IEEE/CVF ICCV*. 1284–1293.
[42] Jieming Yin et al. 2018. Modular routing design for chiplet-based systems. In *2018 ACM/IEEE 45th Annual ISCA*. IEEE.
[43] Shihui Yin et al. 2019. Vesti: Energy-efficient in-memory computing accelerator for deep neural networks. *IEEE TVLSI* 28, 1 (2019), 48–61.
[44] Zhenhua Zhu et al. 2020. MNSIM 2.0: A behavior-level modeling tool for memristor-based neuromorphic computing systems. In *Proceedings of the 2020 on Great Lakes Symposium on VLSI*. 83–88.
[45] Brian Zimmer et al. 2019. A 0.11 PJ/Op, 0.32-128 TOPS, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16nm. In *2019 Symposium on VLSI Circuits*. IEEE, C300–C301.
[46] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

Received April 2021; revised June 2021; accepted July 2021