SWAP: A Server-Scale Communication-Aware Chiplet-Based Manycore PIM Accelerator

Harsh Sharma, *Graduate Student Member, IEEE*, Sumit K. Mandal, *Graduate Student Member, IEEE*, Janardhan Rao Doppa, *Senior Member, IEEE*, Umit Y. Ogras, *Senior Member, IEEE*, and Partha Pratim Pande, *Fellow, IEEE*

***Abstract*—Processing-in-memory (PIM) is a promising technique to accelerate deep learning (DL) workloads. Emerging DL workloads (e.g., ResNet with 152 layers) consist of millions of parameters, which increase the area and fabrication cost of monolithic PIM accelerators. The fabrication cost challenge can be addressed by 2.5-D systems integrating multiple PIM chiplets connected through a network-on-package (NoP). However, server-scale scenarios simultaneously execute multiple compute-heavy DL workloads, leading to significant interchiplet data volume. State-of-the-art NoP architectures proposed in the literature do not consider the nature of DL workloads. In this article, we propose a novel server scale 2.5-D manycore architecture called SWAP that accounts for the traffic characteristics of DL applications. Comprehensive experimental evaluations with different system sizes as well as diverse emerging DL workloads demonstrate that SWAP achieves significant performance and energy consumption improvements with much lower fabrication cost than state-of-the-art NoP topologies.**

***Index Terms*—2.5-D, chiplet, network-on-package (NoP), processing-in-memory (PIM).**

I. Introduction

DEEP learning (DL) workloads, such as deep neural networks (DNNs), convolutional neural networks (CNNs), graph neural networks (GNNs), and their variants, are employed in a range of applications, including autonomous vehicles, medical diagnosis, video analytics, recommendation systems, and social networks [1], [2]. Due to their data-parallel nature, DL workloads are usually accelerated using GPUs. However, general-purpose GPUs lead to suboptimal performance since they are not customized for executing DL workloads. The most notable limitations of GPU architectures are: 1) high power and area overheads; 2) low performance-per-watt; and 3) limited memory bandwidth. To address these limitations, DL hardware accelerators must include domain-specific customizations to optimize performance and energy-efficiency tradeoffs.

Recent research has demonstrated the potential of resistive random-access memory (ReRAM) for efficient DL training and inference [3]. DL computation kernels mainly involve multiply-and-accumulate (MAC) operations, which are efficiently implemented using ReRAM-based architectures. ReRAMs also allow for processing-in-memory (PIM), which helps reduce communication between computing cores and the main memory, increasing energy efficiency. Unfortunately, as the complexity of DL models continues to grow, tiled ReRAM architectures with tens to hundreds and thousands of processing elements (PEs) are required to support efficient training and inference. Monolithic ReRAM implementations, even with 3-D stacking, are no longer viable solutions at this scale due to temperature hotspots and additional dollar costs with the "keep out regions" [4], [5], [6]. In contrast, 2.5-D integration, which involves integrating multiple small chips (i.e., chiplets) on an interposer, represents an effective solution to improve yield and scaling for larger and complex DL models.

Novel 2.5-D chiplet platforms provide a new avenue for compact scale-out implementations of emerging DL workloads. Integrating multiple small chiplets on a large interposer offers significant performance and manufacturing yield improvements compared to 2-D ICs, reducing the fabrication cost [7]. Furthermore, it also achieves higher thermal efficiency than 3-D ICs and facilitates heterogeneous integration [8]. Hence, it has become possible to envision large-scale accelerators on 2.5-D platforms to support the resource-intensive DL applications. However, scalable communication between chiplets is particularly challenging due to relatively large physical distances between chiplets, poor technology scaling of electrical wires, and shrinking power budgets. The aforementioned challenges make it difficult to design viable network-on-package (NoP) that can support ultrahigh bandwidth, energy-efficient, and low-latency interchiplet data transfers without increasing fabrication costs. The demands on the NoP infrastructure will only be exacerbated as application complexity continues to scale. For example, the NoP area overhead alone can be up to 85% of the total system area [9]. However, none of the prior studies have considered specific optimization for NoP while executing DL workloads on chiplet-based systems.

Manuscript received 25 July 2022; accepted 25 July 2022. Date of publication 9 August 2022; date of current version 24 October 2022. This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CNS-1955353; and in part by the Semiconductor Research Corporation Under Task ID 3012.001 and Task ID 3014.001. This article was presented in the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES) 2022 and appears as part of the ESWEEK-TCAD special issue. This article was recommended by Associate Editor A. K. Coskun. *(Corresponding author: Partha Pratim Pande.)*

Harsh Sharma, Janardhan Rao Doppa, and Partha Pratim Pande are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164 USA (e-mail: harsh.sharma@wsu.edu; jana.doppa@wsu.edu; pande@wsu.edu).

Sumit K. Mandal and Umit Y. Ogras are with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI 53715 USA (e-mail: skmandal@wisc.edu; uogras@wisc.edu).

Digital Object Identifier 10.1109/TCAD.2022.3197500

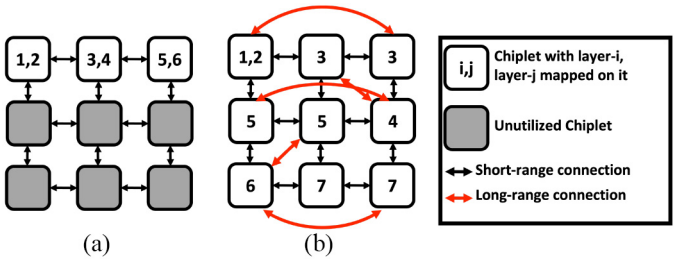

Fig. 1. (a) Mapping of a neural network with small number of parameters leaves many unutilized chiplets, as well as NoP. (b) Mapping of a neural network with large number of parameters increases chiplet utilization, but introduces long range communications.

In this work, we propose design of a high-performance and energy-efficient NoP architecture specifically targeted for DNN inference. To achieve high performance and energy efficiency, we integrate multiple ReRAM chiplets in a single 2.5-D system *via* an optimized NoP. DNNs typically exhibit sequential execution, i.e., computations corresponding to neural layer-n are performed after all data from layer-$(n-1)$ reach layer-n. Then, communication between layer-n and layer-$(n+1)$ will start, and so forth. When the DL workload consists of a limited number of parameters, one chiplet is adequate to execute one or more layers of the DNN. In this case, we do not need to map one neural layer to multiple chiplets. Hence, the interchiplet communication is confined only between the neighboring chiplets executing two consecutive layers due to the aforementioned sequential nature of DNN execution. Fig. 1(a) shows such a scenario where the interchiplet communication happens only between the neighboring chiplets placed on the topmost row. In this situation, most of the chiplets and the NoP remain underutilized while executing the DNN. However, when each layer of the DNN contains a large number of multibit weights (e.g., ResNet-110 on ImageNet with 96.8 M parameters, DenseNet121 on ImageNet with 262.98 M parameters), multiple chiplets are required to map one specific layer. This setting is specifically applicable for server class applications. It should be noted that the Amazon AWS offers instances to run ResNet18, ResNet50, etc. on their cloud infrastructure [32], [33]. Distributing one layer across multiple chiplets increases the volume of interchiplet data transfer. This situation is aggravated further when multiple DL workloads run simultaneously on a server-class system. Hence, when server-class systems execute multiple DL workloads in parallel or one huge DL model alone, most chiplets are highly utilized. In this situation, data exchange is required between chiplets that are physically apart. A similar scenario is depicted in Fig. 1(b), where a single layer is distributed across multiple chiplets. Hence, the interchiplet communication consists of both local and long-range traffic [long red arrows in Fig. 1(b)] in server-class scenarios. Additionally, the long-range traffic pattern varies from one DNN to another. Hence, existing NoP architectures that use straightforward extensions of regular interconnection topologies, like mesh and torus, are not suitable for addressing the interchiplet communication in server-class 2.5-D systems [9], [10], [11].

This article proposes a DL workload-centric NoP design and optimization methodology. Specifically, we develop an efficient multiobjective optimization (MOO) algorithm to generate a NoP architecture where the number of links associated with each router (number of router ports) varies depending on the interchiplet traffic pattern. The irregularity in the generated NoP architecture improves the overall link utilization in the system. The proposed Server-scale Communication-aware Chiplet-based Manycore PIM Accelerator (SWAP) is scalable for a wide variety of DL workloads and the number of chiplets in the system. To demonstrate the versatility of the SWAP architecture, we also present performance evaluation with graph analytics, a significant non-DL application.

Experimental evaluations with a wide range of DL workloads and system sizes show that SWAP simultaneously improves energy consumption, latency, and fabrication cost. Specifically, it achieves up to 42.4% lower energy, 10.6% lower latency, and 10.6× lower fabrication cost than state-of-the-art NoP topologies. The major contributions of this article are as follows.

1) An end-to-end framework that optimizes placements of chiplets and interchiplet links for DL workloads to design an irregular NoP architecture.
2) Demonstrating the applicability of the proposed framework to multiple DL workloads that are executed simultaneously on the 2.5-D system, emulating a server-class architecture.
3) Extensive experimental evaluations that demonstrate performance and energy improvements with significantly lower fabrication costs compared to state-of-the-art NoP architectures.

The remainder of this article is organized as follows. Section II describes the prior work on 2.5-D systems and ReRAM-based PIM targeted for DL workloads. Section III presents the SWAP architecture along with the associated MOO framework. Section IV presents the detailed experimental results and Section V presents the performance evaluation of SWAP with graph analytics workload. Finally, Section VI concludes this article by highlighting the salient contributions.

II. Related Work

This work focuses on accelerating CNN inference on 2.5-D systems with ReRAM-based chiplets. Hence, this section reviews the prior work on 2.5-D architectures and ReRAM-based accelerators for DL workloads.

1) 2.5-D-Based Architectures: The increasing fabrication costs can mask the performance improvement of large monolithic manycore architectures. Most chip vendors and foundries, including TSMC, NVIDIA, Intel, and AMD, are exploring nonmonolithic alternatives such as 2.5-D interposer-based systems to partition the on-chip resources into smaller discrete computing cores called *chiplets*. 2.5-D-based manycore systems offer a promising alternative to monolithic chips [10].

To date, researchers have proposed 2.5-D-based systems for general-purpose, as well as domain-specific applications. IntAct is a 2.5-D prototype system with six 16-tile chiplets

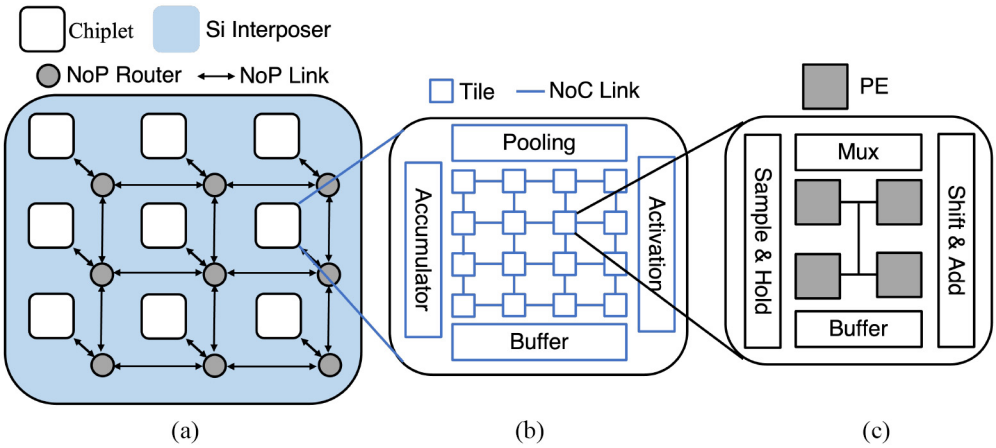

Fig. 2. Chiplet-based PIM architecture utilized within SWAP. The architecture consists of chiplets with their associated NoP routers. Each chiplet consist of 16 tiles with buffer, accumulator, activation unit, and pooling layer. Tiles are connected on a Mesh network.

stacked on an active interposer [6]. In IntAct, the authors demonstrate the scalability of the computing system with low latency interconnects between chiplets. SIMBA is another 2.5-D architecture with 36 chiplets targeting DL applications [11]. Tiling optimizations are introduced to reduce interchiplet communication. However, none of the prior work considers any NoP-specific architectural modifications, which take the workload characteristic into account. A scalable evaluation framework for 2.5-D systems, SIAM is proposed recently. The framework enables fast exploration of various design parameters of a 2.5-D system under DL workloads. Yet, the evaluation framework proposed in SIAM assumes mesh-based NoP topology. Since DL workloads incur irregular traffic patterns, a regular mesh NoP topology may compromise the achievable performance.

Recently, the Kite family of NoP topologies has been proposed for a 2.5-D-based system considering synthetic traffic/workloads [10]. However, Kite is not optimized for DL workloads and is primarily based on a Torus architecture.

2) ReRAM-Based Architectures: ReRAMs can be used both as memory and to perform *in-situ* MAC operations [12], [13]. Since DL workloads rely heavily on such MAC operations, ReRAM-based architectures are excellent candidates for DL training and inference [12], [13], [14]. ReRAM-based heterogeneous architectures were proposed to improve the accuracy of trained models while also addressing communication bottlenecks [15], [16]. Thus, ReRAMs can outperform CPUs/GPUs for almost all types of DL workloads as they support near-data computation. Several recent research studies have devised ReRAM-based DL accelerators that overcome the limited write endurance and high write energy costs of ReRAMs [17], [18].

In this work, we design a chiplet-based 2.5-D architecture, referred to as SWAP. It is enabled by an irregular NoP and leverages the PIM capabilities of ReRAM-based chiplets to accelerate DNN inference on a server-scale system. To the best of our knowledge, SWAP is the first 2.5-D accelerator with interchiplet communication-aware NoP to achieve high performance and energy efficiency with reduced fabrication cost with respect to state-of-the-art alternatives.

III. SWAP Design and Optimization

This section presents the overview and design methodology of the SWAP architecture. We start by showing the overall SWAP architecture, including the intra- and inter-chiplet configurations. We then discuss an efficient MOO methodology to design the overall NoP architecture. It should be noted that the proposed MOO methodology is generic, and it can be used to design other large-scale manycore architectures. This work focuses on the NoP level optimization aspects without modifying the design of individual chiplets.

A. ReRAM-Based SWAP Architecture

Fig. 2 illustrates the underlying 2.5-D design incorporated in SWAP. The top-level architecture, shown in Fig. 2(a), integrates multiple chiplets fabricated on a silicon interposer. The chiplets are connected through NoP routers and links. Fig. 2 uses a mesh topology for NoP as a representative example. Fig. 2(b) shows the structure of a chiplet. Each chiplet consists of 16 tiles along with peripheral circuits, such as accumulator, buffer, activation units (ReLU in our work), and pooling unit. The tiles are connected through a network-on-chip (NoC) [9]. The structure of a tile is shown in Fig. 2(c). Each tile consists of multiple PEs, composed of 128 $\times$ 128 ReRAM crossbar arrays. The crossbars implement the MAC operations. There are 40 PEs inside each tile (only four PEs are shown for brevity) connected with an H-Tree-based point-to-point network [9]. Apart from PEs, each tile consists of standard ReRAM peripheral circuits, such as sample&hold, shift&add, multiplexer, and buffers. In this work, we assume that all DNN weights are transferred to the ReRAM chiplets from the DRAM before performing the DNN inference, which is consistent with the prior work [12], [18], [19].

The DNN inferencing workload under consideration needs to be partitioned and mapped onto chiplets to achieve high chiplet utilization. The number of PEs required for mapping a DNN layer depends on the corresponding kernel size, the number of input and output features, along with the bit precision. The aforementioned parameters, in turn, determine

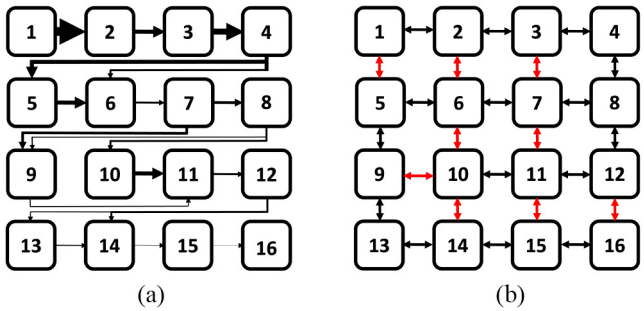

Fig. 3. (a) Traffic exchange between 16 chiplets for a server-scale workload. (b) Connecting chiplets on an SIAM NoP leads to unnecessary links (highlighted in red).

the number of tiles and, hence, the number of chiplets required for each DNN layer. For a server-scale scenario, the number of DNN parameters is in the order of hundreds of millions [20]. We also note that each chiplet cannot be arbitrarily large due to yield and reliability challenges [4]. Hence, it is not always feasible to store parameters of successive DNN layers in neighboring chiplets. Therefore, consecutive DNN layers may be executed by chiplets that are physically far apart. Consequently, it gives rise to long-range traffic in addition to the local traffic between two neighboring chiplets. This leads to a mix of short- and long-range interchiplet data exchange, as depicted in Fig. 3. Figs. 3(a) and (b) show the actual data flow and the traffic in the SIAM(mesh NoP), respectively. In SIAM, the links shown in red do not carry any traffic. Hence, a multistage mesh-based NoP where one chiplet is connected to all of its four neighbors is not necessary and gives rise to multiple redundant links carrying almost no traffic. Hence, constructing an NoP architecture with appropriate placement of chiplets and associated links, which considers interchiplet communication traffic is essential. To this end, we formulate the chiplet placement as well as interchiplet link placement problem to design the overall SWAP architecture as an MOO problem. Chiplet placement and NoP link placement should accommodate the high interchiplet traffic volume prevalent in server-scale applications. The SWAP architecture is optimized to achieve high throughput. Load balancing is a popular way to address this problem [21]. Hence, we compare the degree of achievable load balancing in the NoP designs with different chiplet and link placements using the mean and standard deviation of the traffic (load) on each link [21]. Minimizing the mean and standard deviation of the traffic distribution leads to an overall higher throughput of the candidate SWAP designs.

B. NoP Fabrication Cost Analysis

A 2.5-D system can achieve the functionality of a large chip with a higher yield. However, the cost of the interposer has to be accounted for [4]. In this section, we analyze the fabrication cost of NoP as a function of the number NoP routers and links. The NoP is the biggest contributor to the overall 2.5-D system area [4]. Hence, reducing the NoP area is of paramount importance. The NoP consists of routers and links. The total NoP area (A_{NoP}) is proportional to the sum of the area of the NoP routers and the links [9]

$$A_{\text{NoP}} \propto \left(\sum_{i=1}^{L} A_{\text{router}_i} + \sum_{j=1}^{P} A_{\text{links}_j} \right) \tag{1}$$

where A_{router_i} is the area of ith router and A_{link_j} is the area of the jth link, L and P are the number of NoP routers and links, respectively. Each chiplet is connected to an associated NoP router. So, L denotes the total number of chiplets in the system too. Therefore, increasing the number of router ports (both input and output), as well as NoP links increase the total NoP area. Furthermore, the normalized fabrication cost of an NoP is expressed as [9]

$$C_{\text{NoP}} = \frac{L_{\text{ref}}}{L} \times e^{-D_0(A_{\text{ref}} - A_{\text{NoP}})} \tag{2}$$

where L_{ref} is the number of chiplets per wafer in the reference system and N is the number of chiplets per wafer for the system under consideration. The parameter D_0 represents the wafer defect density, and A_{ref} is the NoP area of the reference system. We consider a 2.5-D system designed by AMD with 864 mm^2 interposer area and 64 chiplets as the reference in this work [4].

Using (2), we are able to compare the fabrication cost of two different NoP architectures. For an example, NoP fabrication cost for SWAP (C_{SWAP}) is

$$C_{\text{SWAP}} = \frac{L_{\text{ref}}}{L} \times e^{-D_0(A_{\text{ref}} - A_{\text{SWAP}})}. \tag{3}$$

Similarly, the fabrication cost of the mesh-based SIAM NoP is

$$C_{\text{SIAM}} = \frac{L_{\text{ref}}}{L} \times e^{-D_0(A_{\text{ref}} - A_{\text{SIAM}})} \tag{4}$$

where A_{SWAP} and A_{SIAM} correspond to total NoP area of SWAP and SIAM, respectively. Therefore, the fabrication cost of SWAP with respect to SIAM can be expressed as

$$\frac{C_{\text{SWAP}}}{C_{\text{SIAM}}} = e^{-D_0(A_{\text{SIAM}} - A_{\text{SWAP}})}. \tag{5}$$

It is evident from (5) that the relative fabrication cost of SWAP and other architectures like SIAM principally boils down to the difference between the two NoP areas. Since the NoP area increases with increasing number of router ports and NoP links, the corresponding fabrication cost also increases. Therefore, underutilized router ports and NoP links lead to unnecessary fabrication cost without any performance gain. Hence, optimizing NoP router and link configuration is crucial.

C. C2F-MOO Methodology for NoP Design

In this section, we first describe the MOO formulation to design the SWAP NoP. Next, we describe the MOO algorithm to improve the accuracy and efficiency of design optimization.

MOO Formulation: Each candidate design d in the NoP design space D corresponds to a specific placement of the chiplets and interchiplet links. To aid the design optimization of NoP, we assume that the communication traffic pattern (F_{ij}, the amount of traffic between chiplets i and j, respectively), is provided as an input to the algorithm. Note that F_{ij} can be obtained by profiling the application workload, i.e., inference on DNN models in our case. Our goal is to achieve high throughput under the given application traffic

Algorithm 1: C2F Design Space Exploration

Input: Number of chiplets, Maximum number of communication links M, communication traffic F_{ij}, multiple objectives $O = \{\mu,\ \sigma\}$

Output: Pareto-optimal set of designs D*

Algorithm:

Initialize: $D^*(M) = MOO_{Solver}(M, O)$

$M' = M$ *and* $\delta = M/10$

Reduce the number of communication links: $M' = M' - \delta$

Repeat

  If $PHV(D^*(M)) > PHV(D^*(M'))$**:**

  Increase the number of NoP links: $M' = M' + \delta/2$ // Finer step

  Else:

  Reduce the number of NoP links: $M' = M' - \delta$ // Coarse step

  End if

  $D^*(M') = MOO_{Solver}(M',\ O)$

Until convergence

TABLE I
SUMMARY OF NOTATIONS USED IN THIS ARTICLE

L	Number of Chiplets/Routers	σ	Standard deviation of the traffic on each NoP link
P	Number of links on the NoP	q_{ijk}	Boolean representation if link k is being used between router i and j
N	Maximum number of links in an all-to-all connected NoP	d_l	Candidate placement of links
M	Number of links in a 2D Mesh	d_c	Candidate placement of chiplets
A	NoP Area	D^*	Optimized pareto set of NoP designs
C	Fabrication Cost normalized to a reference area	E	Evaluation function
F_{ij}	Amount of traffic between chiplet i and chiplet j	δ	Step size while reducing links in C2F-MOO
u_k	Link utilization of link k		
μ	Mean of traffic on each NoP link		

pattern. Minimizing the mean $\mu(d)$ and standard deviation $\sigma(d)$ of the traffic distribution leads to an overall higher throughput of the candidate NoP designs. We can represent the MOO-formulations as follows:

$$D^* = \text{MOO}(\text{objective} = \mu(d),\ \sigma(d)) \tag{6}$$

where D^* is the set of Pareto optimal designs. Here, we aim to find the optimal placement of both chiplets and links such that the design requirements for all the elements are satisfied. A design is called *Pareto optimal* if it cannot be improved in any of the objectives without compromising some other objective. Our goal is to find the Pareto optimal set of NoP designs $D^* \subseteq D$ by using **at most** the same number of communication links as for a 2-D Mesh NoP. Below we describe the key elements of our MOO formulation (Table I summarizes the notation).

1) *Design Variables:* There are two types of design variables for NoP-based chiplet design $d = (d_c, d_l)$, where d_c corresponds to a candidate placement of chiplets and d_l corresponds to a candidate placement of communication links between chiplets.
2) *Constraints:* We have two constraints. First, the NoP should enable data transfer between any two pair of chiplets, i.e., NoP should connect all chiplets with no islands. Second, the number of links used to design NoP should not exceed that in a 2-D mesh. However, as shown in Fig. 3(b), we actually can reduce the number of links compared to a mesh NoP. Hence, our MOO formulation allows us to find NoP designs using a lower number of links than in a mesh NoP.
3) *Objectives:* Our goal is to maximize NoP throughput.

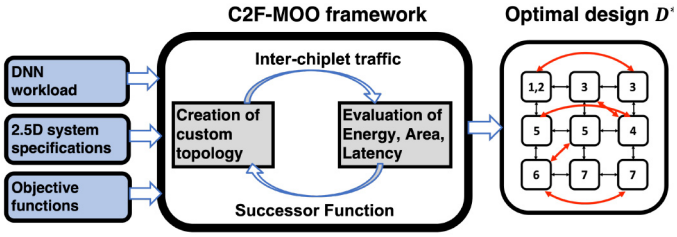

Fig. 4. Illustration of C2F-MOO methodology with design variables and constraints as an input and Pareto optimal designs D^* as output.

Specifically, we consider optimizing two objectives, namely, the mean $\mu(d)$ and standard deviation $\sigma(d)$ of the traffic utilization u_k on each communication link k at time t, determined by the NoP frequency (shown in Table IV later). These parameters are defined below

$$u_k = \sum_{i=1}^{L}\sum_{j=1}^{L} F_{ij}(t) \cdot q_{ijk} \tag{7}$$

$$\mu(d,t) = \frac{1}{P} \times \sum_{k=1}^{P} u_k \tag{8}$$

$$\sigma(d,t) = \sqrt{\frac{1}{P}\sum_{k=1}^{P}(u_k - \mu(d,t))^2} \tag{9}$$

where L represents the number of routers and P represents the number of links in the system as mentioned earlier, q_{ij} is a Boolean representation if link k is being used between router i and router j

$$q_{ijk} = \begin{cases} 1, & \text{if chiplet } i,j \text{ communicate across link } k. \\ 0, & \text{otherwise .} \end{cases}$$

To compute the average throughput across all timestamps, we take a time average of the mean $\mu(d)$ and standard deviation $\sigma(d)$ as represented in (10) and (11)

$$\mu(d) = \text{avg } \mu(d,t) \tag{10}$$

$$\sigma(d) = \text{avg } \sigma(d,t). \tag{11}$$

Once we solve the MOO problem to obtain the Pareto optimal NoP designs $D^* \subseteq D$, we perform cycle-accurate simulations for each design d in D^* to find the design with the lowest EDP and lower latency than a mesh NoP simultaneously. Fig. 4 shows the overview of the C2F-MOO formulation.

ML-Based Coarse-to-Fine Design Optimization: The critical challenge in solving the above MOO problem is the vast combinatorial design space. We need to consider designs with fewer than or equal number of links to a 2-D mesh NoP, which results in a huge combinatorial design space:

$\binom{N}{M}+\binom{N}{M-1}+\binom{N}{M-2}+\ldots$, where N is the number of all possible communication links considering an all-to-all network and M is the number of links in a 2-D mesh NoP (here $P = M$). We propose an efficient algorithm called *Coarse-to-Fine MOO* (C2F-MOO) to address this challenge. C2F-MOO relies on two key ideas. First, we perform a coarse-to-fine design space exploration in terms of the number of allowed communication links: progressively reduce the number of allowed communication links starting at M. Second, we leverage recent advances in machine learning (ML)-based MOO solvers to improve the accuracy and efficiency of solving MOO problems during C2F design space exploration. The critical insight is to extract relevant knowledge to guide the design space exploration toward more promising parts of the design space. Importantly, the learned knowledge can be reused during the C2F design space exploration process to improve the efficiency and accuracy of the overall NoP design optimization.

C2F Design Space Exploration: There are two basic approaches to solve the original MOO problem. First, we can consider all designs with varying number of communications links (M, $M-1$, $M-2$, ...) in a single design space. This poses a significant challenge to effectively explore the design space as any MOO algorithm will require defining a successor function to get a set of neighboring designs for any given design and the size of this set will be huge. Second, we can consider a sequence of MOO problems with the number of communication links equal to M, $M-1$, $M-2$, respectively. We can solve each MOO problem using an existing MOO algorithm in the same order and stop when the quality of the optimized Pareto set of designs (e.g., Pareto hypervolume metric) degrades. In other words, we are taking the smallest possible steps to perform progressive design space exploration. Since solving each MOO problem is expensive and we are solving a large number of MOO problems, this approach is not scalable. Therefore, we propose an efficient coarse-to-fine (C2F) design space exploration approach. We start with the maximum number of links in the 2-D mesh (i.e., M) and solve the MOO problem to obtain the optimized Pareto set of designs. In subsequent iterations, we first take the largest (coarse) step to reduce the maximum number of links (*say by* $\delta = M/10$) and resolve the MOO problem. If the quality of the optimized Pareto set is reduced, we switch to a smaller (finer) step, i.e., the number of reduced links will be less than δ (*say* $\delta/2 = M/20$). Otherwise, we move to the next iteration and start with the largest step δ until convergence.

ML-Based MOO Algorithm: We need an efficient and accurate MOO approach to solve the sequence of MOO problems arising during our C2F design space exploration methodology. Recent work has demonstrated the success of ML-based MOO solvers toward this goal. In this work, we employ the MOO-STAGE algorithm [21]. The key idea behind MOO-STAGE is to learn an evaluation function to select good starting states to guide local search procedures such as greedy search toward better local optima. MOO-STAGE is an iterative algorithm. Each iteration consists of the following three steps. First, we use the current evaluation function to select a good starting state for the local search procedure. Second, we perform a local search from this selected starting state until we reach a local optima and compute the quality of the corresponding Pareto set in terms of Pareto-hyper volume (PHV) as a metric [22]. Third, we update the evaluation function based on the training data from the local search runs. As the evaluation function improves with the training data over iterations, it prunes the design space and directs the search toward more promising regions.

Algorithm 2: ML-Based MOO Solver

Input: Number of chiplets, Maximum number of communication links M, communication traffic F_{ij}, multiple objectives $O = \{\mu,\ \sigma\}$
Output: Pareto-optimal set of designs $D^*(M')$
Algorithm:
1: **Initialize**: evaluation function E, training data D_{train}, and global Pareto set $D^*(M')$
2: **Repeat until Convergence/MAX iterations:**
3: d_{start}= Perform local search guided by E to select the starting state
4: LP = Perform local search from d_{start} to obtain Pareto set
5: Update global Pareto set $D^*(M')$
6: Create new training examples of the form $\{(d_i,\ PHV(LP))\}$
7: Aggregate training data: $D_{train} = D_{train} \cup \{(d_i, PHV(LP))\}$
8: Update evaluation function E: $E = RandomForest(D_{train})$
9: **Return** optimized Pareto set $D^*(M')$

Learning Evaluation Function: The goal of the evaluation function is to estimate the quality of the Pareto set obtained by performing a local search from a given starting design. Given a learned evaluation function, we perform a local search guided by this evaluation function until we reach a local optima to select the starting state (*Meta search*) for the actual local search process (*Base search*). If the evaluation function is accurate, it will avoid bad starting states and select promising starting designs to improve the efficiency and accuracy of design optimization. Otherwise, we update the evaluation function to minimize errors. Each training instance in MOO-STAGE corresponds to the past local search run: sequence of designs (d_1, d_2, ..., d_T) forming the local search trajectory and the quality of the resulting Pareto set from the local optima measured in terms of PHV metric. We create one regression training example for each design d_i on the local search trajectory: d_i is the input and PHV of the local optima is the output. We give the aggregate set of regression examples to the random forest algorithm to create/update the evaluation function. We use random forest as it was shown to be a fast and accurate learner to create effective evaluation functions [23], [24]. Ultimately, our target is to find the optimized design with number of links $M' \leq M$.

IV. Experimental Results

In this section, we present experimental evaluation of the proposed SWAP architecture optimized using our C2F-MOO

TABLE II
LIST OF DL INFERENCE WORKLOADS ALONG WITH THEIR CORRESPONDING NUMBER OF DNN PARAMETERS

Network	#Layers	Dataset	#Parameters (in million)
VGG	16	CIFAR-100	1.67
		ImageNet	38.57
	19	CIFAR100	1.91
		ImageNet	43.8
ResNet	18	CIFAR-100	0.26
		ImageNet	6.06
	34	CIFAR-100	0.51
		ImageNet	22.8
	50	CIFAR-100	0.76
		ImageNet	45.31
	101	CIFAR-100	1.11
		ImageNet	91.01
	110	CIFAR-100	1.69
		ImageNet	96.8
	152	CIFAR-100	2.36
		ImageNet	129.11
DenseNet	40	CIFAR-100	1.6
	121	ImageNet	262.98
	169	ImageNet	892.72

methodology on emerging DL workloads and compare it to prior NoP designs for chiplet platforms.

A. Experimental Setup

Datasets and DL Workloads: We evaluate the SWAP architecture on a wide range of DNNs for inferencing. Table II shows different DNNs, corresponding datasets, and the number of parameters. Note that, the DNNs considered in our evaluations consist of linear (VGG), residual (ResNet), as well as dense (DenseNet) connections. Moreover, all the DNNs comprise of both fully connected and convolution layers. The evaluation with various DNNs demonstrates the broad applicability of SWAP.

System Specification and Evaluation Setup: Nonvolatile memory (NVM)-based PIM architectures have been explored extensively [25]. Crossbar arrays (CBAs) are by far the most popular representation for PIM. They are extremely efficient for matrix-vector multiplication, which forms the core of many DL and scientific computing algorithms. Binary CBAs based on phase change memory (PCM), ReRAM, spin-transfer torque memory (STT-MRAM), and ferroelectric field effect transistor memory (FeFETs) have all been studied and experimentally demonstrated at various scales [26], [27], [28]. In this work, we employ ReRAM-based chiplets as the enabling technology to accelerate DNN inference, noting that the SWAP architecture and associated design optimization methodologies are applicable for other CBA-based PIM chiplets as well. Moreover, our evaluations use the ReRAM technology due to its energy and area benefits. However, SWAP can also be used with other memory technologies such as SRAM. To demonstrate the scalability of the SWAP architecture, we consider three different system sizes with 36, 64, 81 chiplets. Each system is capable of executing one large or more than one DL workloads simultaneously representing server-scale scenarios. We use a modified NeuroSim engine to partition and map DL workloads on to a 2.5-D-based system while estimating the number of chiplets required [29]. The interchiplet traffic (F_{ij}) is generated by the activations between the chiplets. Each chiplet covers about 2.64mm^2 area, including the peripherals. The C2F-MOO framework results in design D^* that defines the chiplet and link placements in the SWAP architecture. All the NoP topologies are simulated using the BookSim simulator [30]. The inputs to the BookSim simulator are the connectivity between NoP routers and the interchiplet traffic for DL workloads. It outputs the area, latency, and energy consumption of the NoP. We use the Nvidia ground-referenced signaling (GRS) parameters for chiplets on a 32-nm technology to evaluate the NoP area and power consumption. Tables III(a)–(c) show the average chiplet utilization for the 36-, 64-, and 81-chiplet systems while executing the DL workloads. The DL workloads either consist of one large model or multiple models are executed simultaneously. It should be noted that for all the cases the chiplets are highly utilized, mostly 90%–100%. The only exception is the workload of ResNet 18, 34, 50, 110, 152 running simultaneously on CIFAR-100 for the 36-chiplet system where the average utilization is 72%. Even when all the above mentioned ResNet workloads are executed with CIFAR-100 dataset then the total number of parameters is less than any other configuration. Hence, in this situation, the chiplet utilization is lower than the others. Table IV shows the other system-level parameters considered in the performance evaluation [6], [31]. We note that SWAP will achieve similar performance gains with other technology parameters.

Baseline Chiplet Designs: We compare SWAP against two baseline designs: 1) Kite [10] and 2) SIAM [9]. Kite is a Torus-based NoP that employs skip connections along with shorter links, maintaining a concentration factor of four. We incorporate the Kite-medium topology as this is the best topology among all the Kite topologies. SIAM principally is a mesh NoP. We set same system parameters for all three architectures (Kite, SIAM, and SWAP) for a fair comparison. All the bar plots are normalized with respect to SIAM.

B. Variation in Number of Router Ports

SWAP consists of NoP routers with varying number of ports (in contrast to a uniform distribution) determined by the C2F-MOO framework, as described in Section III. In this section, we compare the distribution of the number of router ports in the SWAP architecture with respect to the baseline designs. Fig. 5(a) shows the comparison for a 36-chiplet system. We observe that five-port routers are the most frequent one with Kite. SIAM with mesh NoP mostly consists of routers with four ports. In contrast, SWAP mostly uses two-port routers. The peak moves toward the left, demonstrating that the frequency of routers with lesser number of ports is increasing in the case of SWAP. Similarly, as the system scales to 64- and 81-chiplets, both Kite and SIAM have a average port count of around four as shown in Figs. 5(b) and (c), respectively. In case of SWAP, the peak moves toward left with mean router port frequency being between two and three.

TABLE III
CHIPLET UTILIZATION IN PERCENTAGES FOR (A) 36 CHIPLETS (CIFAR-100), (B) 64 CHIPLETS (IMAGENET), (C) 81 CHIPLETS (IMAGENET)

Network (CIFAR-100)	Average Chiplet utilization (In percentage)
VGG19	100%
VGG16	91%
ResNet18,34,50, 110,152	72%
VGG16, ResNet18	97%
VGG16, ResNet34	100%
VGG16, DenseNet40	100%

(a)

Network (ImageNet)	Average Chiplet utilization (In percentage)
VGG19, ResNet18	93.75%
ResNet18,34,50	90.65%
ResNet18,101	92.1%
ResNet110	92.1%
DenseNet121	90%

(b)

Network (ImageNet)	Average Chiplet utilization (In percentage)
VGG19, ResNet50	91.3%
ResNet101,34	92.5%
ResNet101,50	90.1%
ResNet152	92.5%
DenseNet169,ResNet50,18	98.7%

(c)

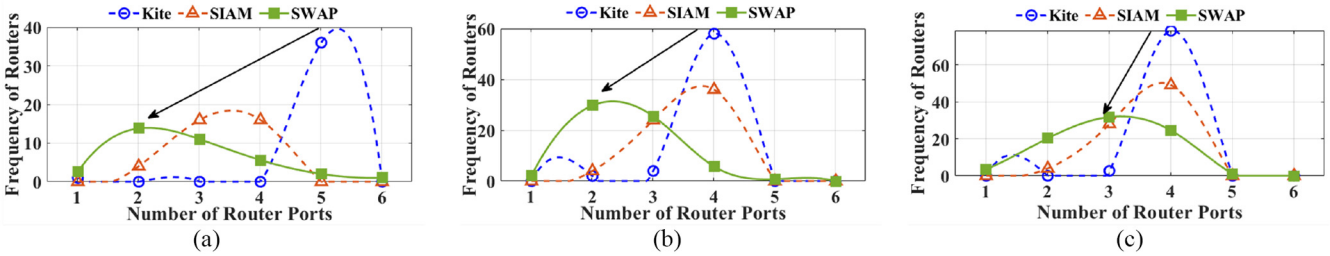

Fig. 5. Router port configuration for Kite, SIAM, and SWAP for a 2.5-D system with (a) 36 chiplets, (b) 64 chiplets, and (c) 81 chiplets. Peak of the plot is observed to be moving toward left in case of SWAP.

TABLE IV
NOP HARDWARE PARAMETERS CONSIDERED FOR EVALUATION

NoP frequency	**1.15 GHz**
NoP bus width	**32**
One hop NoP link length	**1.449mm**
Quantization bit	**8**
Technology	**32nm**
Link frequency	**0.6ns/mm**

SWAP mainly consists of routers with lower number of ports as the C2F-MOO framework described in Section III improves link utilization, and removes unnecessary NoP links. Hence, number of router ports reduces too. Smaller routers in SWAP helps in reducing NoP energy, area, and fabrication cost, as we show in the following sections.

C. NoP Performance and Energy Analysis

This section presents the NoP latency and energy for SWAP and the baseline designs (Kite and SIAM). Figs. 6(a)–(c) show the NoP latency for all the three NoP architectures under consideration. We observe that SWAP outperforms both the baselines for all the system sizes. For the 36-chiplet SWAP, the highest latency improvement of 5% is achieved for VGG19. Similarly, SWAP shows 8% and 11% improvements in latency with 64 and 81 chiplets, respectively.

SWAP not only reduces the inference latency of DL workloads but also achieves significant energy consumption reduction. The energy consumption improvements compared to Kite and SIAM are shown in Figs. 7(a)–(c). SWAP achieves on average 36%, 45%, and 47% lower energy than Kite for 36, 64- and 81-chiplet-based systems, respectively. Similarly, we observe on an average 11%, 22%, and 25% lower energy than SIAM for 36, 64, and 81 chiplets, respectively. As expected, simultaneous latency and energy improvements lead to significant EDP benefits, as shown in Figs. 8(a)–(c). On average, SWAP achieves 35%, 51%, and 62% lower EDP than Kite with 36, 64, and 81 chiplets, respectively. Moreover, SWAP shows 11%, 25%, and 20% improvement in EDP over SIAM with 36, 64, and 81 chiplets, respectively. Fig. 9 compares the general trend in latency and energy between SWAP and baseline designs for the system with 81 chiplets. SWAP significantly improves both latency and energy for 3 out of 4 server-scale workloads with respect to Kite, as shown in Fig. 9(a). Although the workload consisting of VGG19 and ResNet50 shows little latency improvement with SWAP, it shows a considerable improvement in energy. Fig. 9(b) shows that SWAP achieves 5% to 7% improvement in energy over SIAM without sacrificing latency. While running DenseNet169, ResNet18, and ResNet50 together, SWAP achieves over 62% and 40% improvement in energy over Kite and SIAM, respectively. Both Kite and SIAM incorporate regular NoP topologies and consist of several links which are not necessary for DL workloads. In contrast, SWAP consists of an optimized NoP that removes redundant links and places them appropriately based on interchiplet communication traffic. In summary, smaller routers and fewer appropriately placed links enable SWAP to achieve lower latency and energy efficiency than both Kite and SIAM.

D. Application-Agnostic NoP Design

The SWAP architecture, along with the NoP optimized for one type of DL application, shows similar performance for other DL applications as well. We show

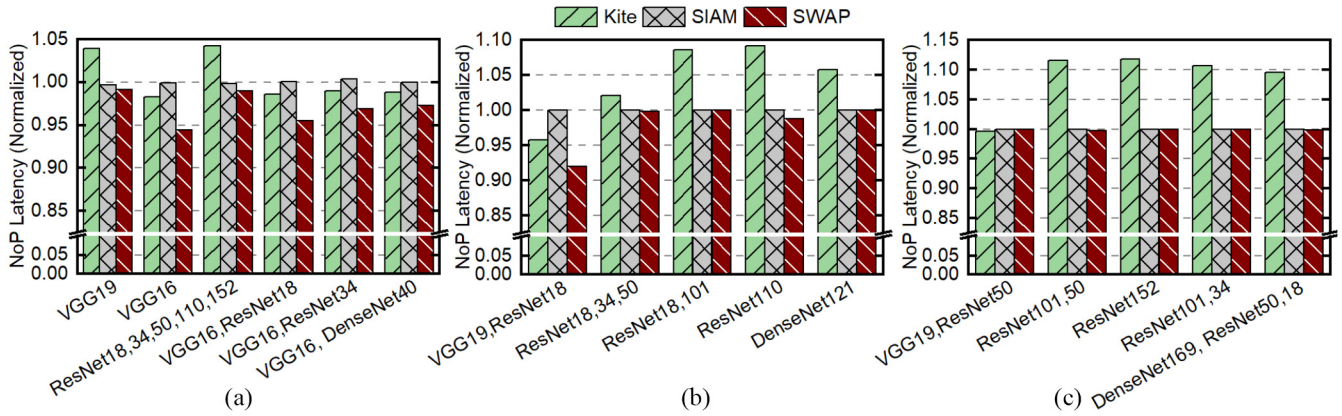

Fig. 6. Comparison of NoP latency for 2.5-D system with (a) 36 chiplets, (b) 64 chiplets, and (c) 81 chiplets.

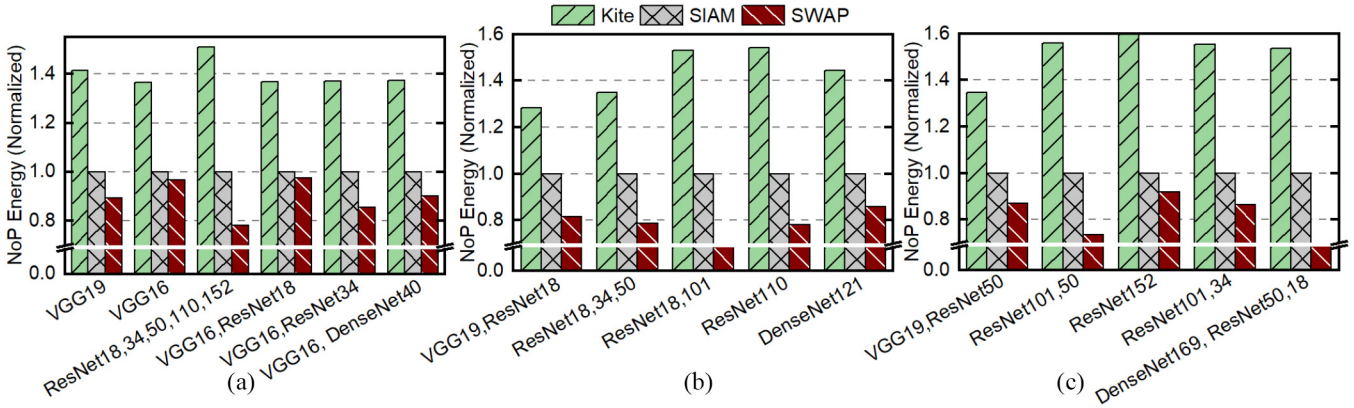

Fig. 7. Comparison of NoP energy for 2.5-D system with (a) 36 chiplets, (b) 64 chiplets, and (c) 81 chiplets.

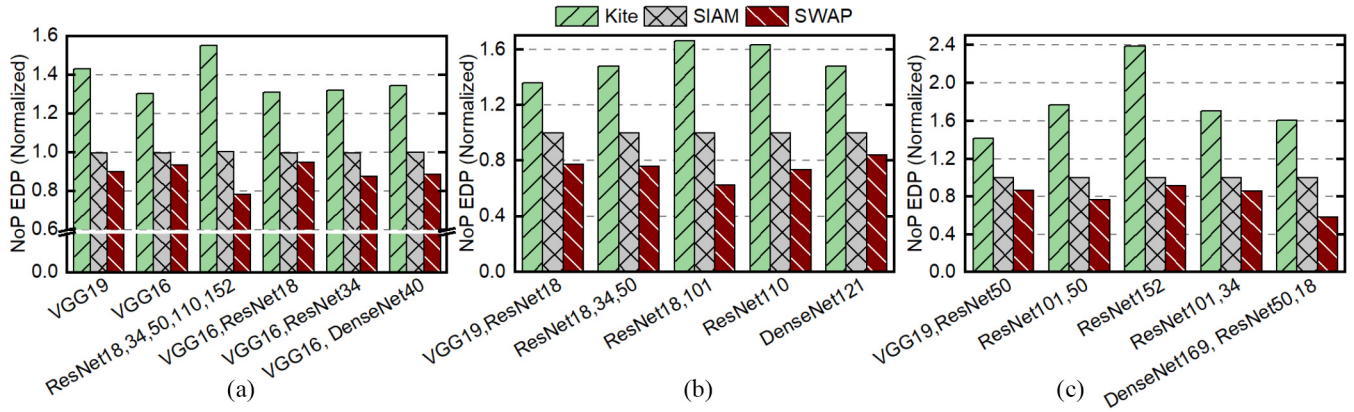

Fig. 8. Comparison of NoP EDP for 2.5-D system with (a) 36 chiplets, (b) 64 chiplets, and (c) 81 chiplets.

the application-agnostic nature of SWAP by evaluating the performance of a 36 chiplet-based system for different DL workloads with similar chiplet utilization shown in Table V. Here, we consider four workloads, denoted by WL1, WL2, WL3, and WL4. Hence, to design an application-agnostic NoP, we generate an NoP optimized for each DL workload (denoted by the specific application). The optimized NoPs are then used to execute all the other DL workloads, e.g., an NoP optimized for WL1 is used to execute the other three DL workloads (WL2, WL3, and WL4) and the performance is normalized with respect to the application-specific NoP. Fig. 10 shows the normalized NoP latency for the SWAP architecture with 36 chiplets. We see only 0.7% degradation on average for all applications when compared to application specific NoPs with the worst-case degradation reaching only up to 3.6%.

TABLE V
LIST OF DL WORKLOADS ALONG WITH THEIR CHIPLET UTILIZATION

DL workload	Chiplet utilization
VGG19 (WL1)	100%
VGG16,ResNet18 (WL2)	97 %
VGG16,ResNet34 (WL3)	100%
VGG16,DenseNet40 (WL4)	100%

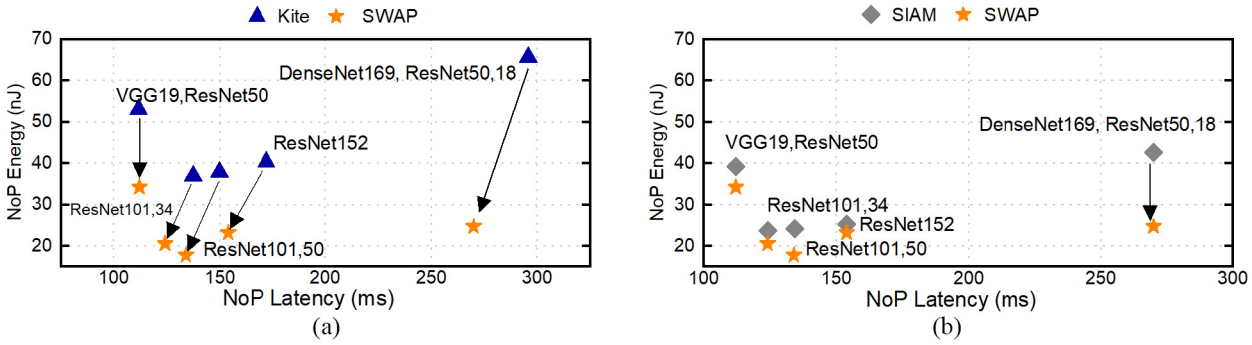

Fig. 9. Trend in NoP energy and latency for (a) Kite and SWAP; (b) SIAM and SWAP for a 2.5-D system with 81 chiplets.

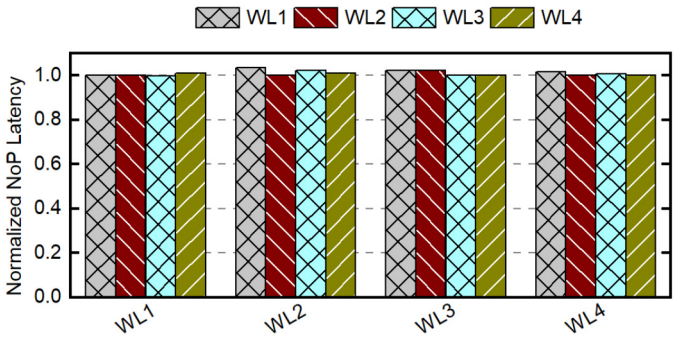

Fig. 10. Normalized NoP latency of 36-chiplet SWAP architecture to study the performance degradation with respect to application-specific designs. The results show that SWAP is versatile to multiple applications.

TABLE VI
LIST OF GRAPH-BASED DATASETS ALONG WITH THEIR CORRESPONDING NUMBER OF NODES AND EDGES

Dataset (Input Graph)	Number of nodes	Number of edges
Deezer (DZ)	41k	125k
Road_luxembourg-osm (RM)	114k	120k
Orkut (OR)	2.9M	20M
FaceBook Anon (FB)	3.1M	23.7M
LiveJournal (LJ)	4.8M	68M

E. Fabrication Cost Analysis

One of the main advantages of 2.5-D systems over monolithic architectures for large-scale designs is the fabrication cost as the system requirement scales. Therefore, it is crucial to compliment the low fabrication cost of 2.5-D systems with performance and energy benefits in a server-scale application. Thus, this section discusses the relative fabrication cost improvement by SWAP with respect to Kite and SIAM. We observe from (1) that the area of the NoP (A_{NoP}), and, hence, the fabrication cost of NoP is directly proportional to the total area occupied by the routers (A_{router}) as well as the links (A_{links}). Figs. 5(a)–(c) show that the average number of router ports is the highest with Kite and the lowest with SWAP. Since router area, as well as link area increase with increasing number of router ports, we can express the NoP router, as well as the link area of Kite, SIAM, and SWAP in the order below

$$(A_{\text{router}})_{\text{Kite}} > (A_{\text{router}})_{\text{SIAM}} > (A_{\text{router}})_{\text{SWAP}}$$
$$\& \ (A_{\text{link}})_{\text{Kite}} > (A_{\text{link}})_{\text{SIAM}} > (A_{\text{router}})_{\text{SWAP}}.$$

Furthermore, from (5), we know that the fabrication cost increases with increasing area. Therefore, it is evident that fabrication cost will follow the same trend as the NoP router and link area. However, the fabrication cost should not be the only metric to compare the practicality of different NoP architecture; it should also be combined with performance metrics. Therefore, Fig. 11 compares the trend in fabrication cost and EDP for SWAP versus Kite and SWAP versus SIAM for the 81-chiplet system. It should be noted that the fabrication cost is the highest for the biggest system size. Hence, we consider the 81-chiplet system for this analysis. We observe that, for all DL workloads, SWAP reduces both EDP and fabrication cost with respect to Kite and SIAM. For example, SWAP shows 57% improvement in EDP combined with a 13X reduction in fabrication cost with respect to Kite while executing ResNet101 and ResNet50 simultaneously, as shown in Fig. 11(a). As shown in Fig. 11(b), SWAP decreases the fabrication cost compared to SIAM by 17X with upto 63% EDP improvement. Therefore, our proposed SWAP architecture with optimized NoP design exhibits high energy-efficiency along with significant reduction in fabrication cost compared to state-of-the-art NoP for 2.5-D systems and demonstrates scalability.

V. SWAP PERFORMANCE EVALUATION WITH GRAPH ANALYTICS WORKLOAD

ReRAM-based PIM modules offer an effective way to address the high memory bandwidth requirement of graph analytics applications by integrating the computing logic in the memory. The ReRAM crossbars can store the adjacency matrix of a graph and the computation in most graph primitives can be decomposed into MAC operations, which are supported by ReRAM. Hence, we evaluate the performance of the SWAP architecture for graph analytics workloads. For this study, we consider the PageRank application with the graph datasets shown in Table VI [34], [35]. We implement PageRank with these 5 datasets on a 64-chiplet system. We also use the C2F-MOO strategy to generate the NoP architecture for the graph application. Fig. 12 compares the trend in EDP and fabrication cost between SWAP and baseline designs for the system with 64 chiplets as an example. SWAP significantly improves both EDP and fabrication cost for all the graph workloads

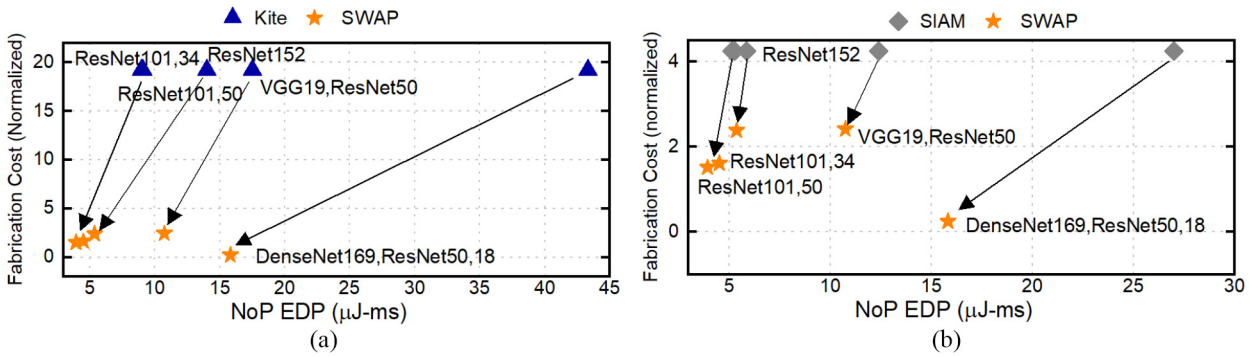

Fig. 11. Trend in fabrication cost and EDP for (a) Kite and SWAP; (b) SIAM and SWAP for a 2.5-D system with 81 chiplets.

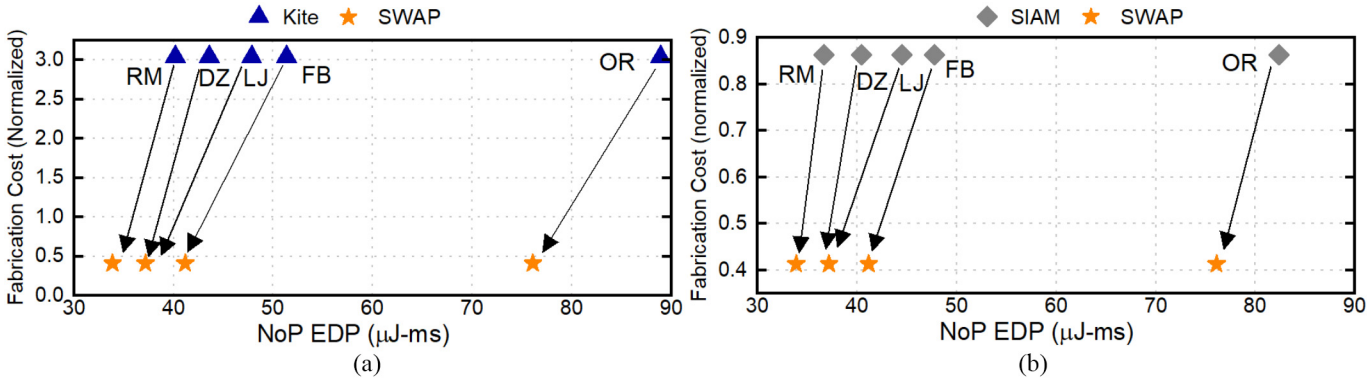

Fig. 12. Trend in fabrication cost and EDP for (a) Kite and SWAP; (b) SIAM and SWAP considering the PageRank application with various graph datasets.

under consideration with respect to Kite and SIAM, as shown in Figs. 12(a) and (b), respectively. Fig. 12 shows that SWAP achieves over 20% improvement in EDP and over 7× improvement in fabrication cost with respect to Kite. We do not repeat the same experiment for other system sizes due to brevity.

VI. Conclusion

This article presented a 2.5-D manycore architecture consisting of ReRAM-based chiplets called SWAP that utilizes an optimized NoP. SWAP uses an ML-inspired coarse-to-fine design optimization approach to integrate multiple chiplets in a 2.5-D system. SWAP is optimized for executing multiple DL workloads simultaneously, representing a server-scale environment. We demonstrated that SWAP outperforms the state-of-the-art 2.5-D manycore architectures with significantly lower energy consumption and fabrication cost. It achieves up to 10.5% lower latency, 42.5% less energy with 13× lower fabrication cost than recently proposed NoP architectures. Optimization of NoP router and link configurations *via* an ML-based design space exploration strategy is the key to SWAP's benefits over its counterparts.

References

[1] Z. Wu *et al.*, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[2] W. Liu *et al.*, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[3] S. Mittal, "A survey of ReRAM-based architectures for processing-in-memory and neural networks," *Mach. Learn. Knowl. Extract.*, vol. 1, no. 1, p. 75–114, 2019.

[4] A. Kannan, N. E. Jerger, and G. Loh, "Enabling interposer-based disintegration of multi-core processors," in *Proc. 48th Int. Symp. Microarchit. (MICRO)*, New York, NY, USA, 2015, pp. 546–558.

[5] N. E. Jerger, A. Kannan, Z. Li, and G. H. Loh, "NoC architectures for silicon interposer systems: Why pay for more wires when you can get them (from your interposer) for free?" in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchit.*, Cambridge, U.K., 2014, pp. 458–470.

[6] P. Vivet *et al.*, "IntAct: A 96-core processor with six chiplets 3D-stacked on an active interposer with distributed interconnects and integrated power management," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 79–97, Jan. 2021.

[7] D. Stow, I. Akgun, R. Barnes, P. Gu, and Y. Xie, "Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5D/3D integration," in *Proc. Int. Conf. Comput.-Aided Des.*, Austin, TX, USA, 2016, pp. 1–6.

[8] X. Zhang *et al.*, "Heterogeneous 2.5D integration on through silicon interposer," *Appl. Phys. Rev.*, vol. 2, no. 2, 2015, Art. no. 21308.

[9] G. Krishnan *et al.*, " SIAM: Chiplet-based scalable in-memory acceleration with mesh for deep neural networks," *ACM Trans. Embedded Comput. Syst*, vol. 20, no. 5, pp. 1–24, 2021.

[10] S. Bharadwaj, J. Yin, B. Beckmann, and T. Krishna, "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *Proc. 57th ACM/IEEE Des. Autom. Conf. (DAC)*, 2020, pp. 1–6.

[11] Y. Shao *et al.*, "Simba: Scaling deep-learning inference with multi-chip-module-based architecture," in *Proc. 52nd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO)*, New York, NY, USA, 2019, pp. 14–27.

[12] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic," in *Proc. ISCA*, 2016, pp. 14–26.

[13] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit. (HPCA)*, Austin, TX, USA, 2017, pp. 541–552.

[14] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. Int. Symp. Comput. Archit.*, Seoul, South Korea, 2016, pp. 27–39.

[15] M. Giordano *et al.*, "CHIMERA: A 0.92 TOPS, 2.2 TOPS/W edge AI accelerator with 2 MByte on-chip foundry resistive RAM for efficient training and inference," in *Proc. IEEE Symp. VLSI Circuits*, 2021, pp. 1–2.
[16] B. Li *et al.*, "3D-ReG: A 3D ReRAM-based heterogeneous architecture for training deep neural networks," *J. Emerg. Technol. Comput. Syst.*, vol. 16, no. 2, pp. 1–24, 2020.
[17] W. Wen, Y. Zhang, and J. Yang, "ReNEW: Enhancing lifetime for ReRAM crossbar based neural network accelerators," in *Proc. IEEE 37th Int. Conf. Comput. Des. (ICCD)*, 2019, pp. 487–496.
[18] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "FloatPIM: In-memory acceleration of deep neural network training with high precision," in *Proc. 46th Annu. ISCA*, 2019, pp. 802–815.
[19] Y. Kim, W. Yang, and O. Mutlu, "RAMULATOR: A fast and extensible DRAM simulator," *IEEE Comput. Archit. Lett.*, vol. 15, no. 1, pp. 45–49, Jan.–Jun. 2015.
[20] B. Zimmer *et al.*, "A 0.32–128 TOPS, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 920–932, Apr. 2020.
[21] B. Joardar *et al.*, "Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems," *IEEE Trans. Comput.*, vol. 68, no. 6, pp. 852–866, Jun. 2019.
[22] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, vol. 63. Aachen, Germany: Shaker, 1999.
[23] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[24] L. Breiman, J. Friedman, R. A. Olshen, and C. Stone, *Classification and Regression Trees*. Pacific Grove, CA, USA: Wadsworth, 1984.
[25] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.
[26] K. Roy, I. Chakraborty, M. Ali, A. Ankit, and A. Agrawal, "In-memory computing in emerging memory technologies for machine learning: An overview," in *Proc. 57th ACM/EDAC/IEEE Des. Autom. Conf. (DAC)*, 2020, pp. 1–6.
[27] W.-H. Chen *et al.*, "CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors," *Nat. Electron.*, vol. 2, pp. 420–428, Aug. 2019.
[28] G. Karunaratne, M. L. Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nat. Electron.*, vol. 3, pp. 327–337, Jun. 2020.
[29] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *Proc. Int. Electron Devices Meeting (IEDM)*, 2019, pp. 1–4.
[30] N. Jiang *et al.*, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. IEEE ISPASS*, 2013, pp. 86–96.
[31] *Intel Foveros Interconnect*, Intel, Santa Clara, CA, USA, 2019.
[32] "Use Apache MXNet (Incubating) for Inference With a ResNet 50 Model." [Online]. Available: https://docs.aws.amazon.com/dlami/latest/devguide/tutorial-mxnet-inference-resnet50.html (Accessed: May 2022).
[33] "Product Overview." [Online]. Available: https://aws.amazon.com/marketplace/pp/prodview-rte234xioxzqu (Accessed: May 2022).
[34] "Stanford Network Analysis Project." [Online]. Available: http://snap.stanford.edu (Accessed: May 2022).
[35] "Network Repository." [Online]. Available: http://networkrepository.com (Accessed: May 2022).

Harsh Sharma (Graduate Student Member, IEEE) received the B.E. degree from NSIT, Delhi University, New Delhi, India, in 2021. He is currently pursuing the Ph.D. degree in computer engineering with Washington State University, Pullman, WA, USA.

His current research interests include analysis and design of chiplet-based architecture, machine learning, and PIM hardware design.

Sumit K. Mandal (Graduate Student Member, IEEE) received the dual (B.Tech.+M.Tech.) degrees from the Indian Institute of Technology Kharagpur, Kharagpur, India, in 2015, and the Ph.D. degree in electrical engineering from the University of Wisconsin–Madison, Madison, WI, USA, in 2022.

His research interests include analysis and design of NoC architecture, AI hardware, and power management of multicore processors.

Dr. Mandal received numerous awards including best paper award from ACM TODAES in 2021.

Janardhan Rao Doppa (Senior Member, IEEE) received the Ph.D. degree in computer science from Oregon State University, Corvallis, OR, USA, in 2014.

He is the Huie-Rogers Endowed Chair Associate Professor in computer science with Washington State University, Pullman, WA, USA. His research interests are at the intersection of machine learning and computing systems design.

Dr. Doppa won NSF CAREER Award, the Outstanding Paper Award from AAAI conference in 2013, the Best Paper Award from ACM TODAES in 2021, the Outstanding Junior Faculty in Research Award in 2020, and the Reid-Miller Teaching Excellence Award in 2018 from the College of Engineering, Washington State University.

Umit Y. Ogras (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2007.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Wisconsin–Madison, Madison, WI, USA. His research interests include embedded systems, heterogeneous systems-on-chip, low-power VLSI, wearable computing, and flexible hybrid electronics.

Dr. Ogras received the DARPA Director's Fellowship Award in 2020, the DARPA Young Faculty Award in 2018, the NSF CAREER Award in 2017, and the best paper awards at 2019 CASES, 2017 CODES+ISSS, 2012 IEEE TCAD, and 2011 IEEE VLSI Transactions.

Partha Pratim Pande (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2005.

He is a Professor and the Holder of the Boeing Centennial Chair in computer engineering with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, USA, where he is currently the Director. His current research interests are novel interconnect architectures for manycore chips, on-chip wireless communication networks, heterogeneous architectures, and ML for EDA.

Dr. Pande has won the NSF CAREER Award in 2009. He is the winner of the Anjan Bose Outstanding Researcher Award from the College of Engineering, Washington State University in 2013. He currently serves as the Editor-in-Chief of IEEE DESIGN AND TEST. He is on the editorial boards of several journals. He was the Technical Program Committee Chair of IEEE/ACM Network-on-Chip Symposium 2015 and CASES (2019–2020). He also serves on the program committees of many reputed international conferences.