A Dataflow-aware Network-on-Interposer for CNN Inferencing in the Presence of Defective Chiplets

Harsh Sharma, *Student Member, IEEE,* Umit Ogras, *Senior Member, IEEE,* Ananth Kalyanraman, *Senior Member, IEEE,* and Partha Pratim Pande, *Fellow, IEEE*

Abstract— **The emergence of 2.5D chiplet platforms provides a new avenue for compact scale-out implementations of deep learning (DL) workloads. Integrating multiple small chiplets using a Network-on-Interposer (NoI) offers not only significant cost reduction and higher manufacturing yield than 2D ICs but also better energy efficiency and performance. However, defects in chiplets may compromise performance since they restrict the computing capability. Therefore, carefully designed chiplet and NoI link placement, and task mapping schemes, in presence of defects, are necessary. In this paper, we propose a defect-aware NoI design approach using a custom-defined space-filling curve (SFC) for efficient execution of mixed workloads of convolutional neural network (CNN) inference tasks. We demonstrate that the *k-ary n-cube*-based NoI topologies can be degenerated into SFC-based counterparts, which we refer to as *SFCed* NoI topologies. They enable high performance and energy efficiency with lower fabrication costs over their parent *k-ary n-cube* counterparts. The SFCed approach helps us to extract high performance from an inherently defective system. We demonstrate that SFCed design achieves up to 2.3× and 3.5× reduction in latency and energy, respectively, compared to parent NoI architectures while executing diverse DL workloads.**

Keywords—2.5D, Chiplet, NoI, Defects, Space filling Curves, CNN Inference, chiplet placement, task mapping

I. Introduction

Recent emergence of complex compute- and data-intensive applications (e.g., autonomous driving, machine vision, robotic medical diagnosis) necessitate high performance with a small form-factor [1] [2]. ITRS 2.0 and IRDS roadmap highlight the unprecedented need for memory and processing over the next decade [3] [4] [5]. This need dictates the design of large-scale chips with high memory and compute capability offering high degrees of parallelism. Such large-scale chips include multiple processing cores, scaling from a few tens to even hundreds. This level of integration increases the area of a monolithic chip significantly [4]. As the monolithic chips approach the reticle limit, exploding fabrication costs rise as one of the major challenges in the silicon industry [6]. Chiplet-based designs that integrate multiple smaller chips (chiplets) on a single interposer offer a promising solution for reducing the manufacturing cost. Since each chiplet occupies a smaller area than a monolithic chip, the overall fabrication cost of the entire 2.5D system is significantly lower than the monolithic counterpart [6]. Then, the chiplets are connected through a network-on-interposer (NoI).

Chiplet-based systems can lower fabrication costs significantly, but realistic design scenarios must consider the impact of defects on the overall performance [4]. Certain parts of individual chiplets may not be functional due to intrinsic defects. However, *none of the prior work considers defects while designing a chiplet-based system*. When a chiplet has defect(s), the standard remedy is disabling the impacted segment and using the chiplet with reduced functionality. Hence, one may need additional chiplet(s) to execute a particular computation kernel (e.g., layers of a deep neural network). This solution, however, increases the inter-chiplet data exchange as activations from one neural layer would have to be spread across multiple chiplets, increasing latency and diminishing performance. Hence, it is critical to design the NoI by considering the impact of defects and their distribution in the chiplets. The number of defects can vary from chiplet to chiplet following well-known probability distributions (discussed in Section III). As interposers in 2.5D systems are 80% empty, the defects on the interposer are negligible [7]. However, no existing NoI architecture considers defects in chiplets while benchmarking achievable performance. Moreover, if the NoI architecture is not optimized for the dataflow generated by the workload, it can aggravate the overall performance degradation.

In this work, we propose a new NoI architecture design technique in the presence of defective chiplets. This design targets CNN workloads on cloud-scale platforms, where the number of parameters can reach billions. In a typical target workload, multiple CNN inference tasks need to be concurrently executed (e.g., object detection, scene understanding in self-driving cars, augmented/virtual reality) on a cloud [8] [9]. Since each neural layer of a CNN typically sends data to the subsequent layer (i.e., the dataflow graph is mostly linear), it is desirable for consecutive neural layers to be mapped to neighboring chiplets.

Existing NoI architectures are primarily based on standard multi-stage regular topologies such as mesh and torus. [10] [11] [12] [13]. Several of these NoI topologies fall under the broad family of *k-ary n-cube* architectures, where *k* is the index and *n* is the dimension of the cube. The *k-ary n-cube* topology represents a method of structuring and interconnecting a network of nodes in a multi-dimensional grid. The more traditional topologies, such as a hypercube and mesh/torus, represent special cases of the *k-ary n-cube* family of topologies. These NoI architectures, however, do not guarantee contiguously placed chiplets to map successive neural layers – which is essential to meet the demands of CNN workloads. Additionally, when one neural layer needs to be mapped on multiple chiplets (due to resource constraints imposed by defects), a multi-stage NoI can potentially increase inter-chiplet traffic, thereby degrading performance.

Dataflow awareness: To overcome these challenges, the proposed NoI architecture connects the chiplets in a contiguous path so that the communicating neural layers are highly likely to span neighboring chiplets without introducing significant long-range and multi-hop data exchange. Our custom-defined *space-filling curve (SFC) enables this NoI design*. SFCs [14] [15] are generic spatial abstractions that help linearize multi-dimensional space. We decompose the parent NoI that was originally designed using some variant of a *k-ary n-cube* topology, into an SFC-based (“*SFCed*”) architecture, enabling a contiguous Hamiltonian path between all the chiplets. This new SFCed representation becomes the network substrate for the chiplets on the NoI, on to which incoming neural layers associated with CNN

This work was supported, in part by the US National Science Foundation (NSF) under grants CNS-1955353, CSR-2308530, CCF 1919122, 2316160.

Harsh Sharma, Ananth Kalyanraman and Partha Pratim Pande are with Washington State University, Pullman, WA, USA. Email: {harsh.sharma, ananth, , pande}@wsu.edu

Umit Ogras is with the Department of Electrical and Computer Engineering, University of Wisconsin Madison, Madison, WI, USA

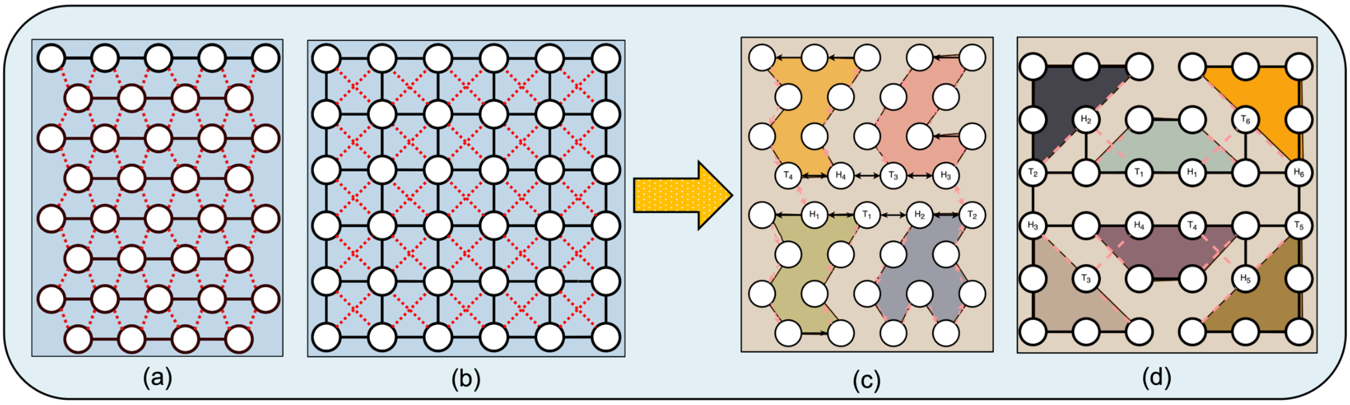


Fig. 1: Illustration of a 36-chiplet-based system with link placement search space for parent (a) HexaMesh NoI (b) Kite NoI; Our degenerated SFC-architecture of (c) HexaMesh NoI with four SFC; (d) Kite NoI with six SFC. Chiplets covered by the same color belong to the same SFC, and the <H, T> denote the head and tail of each corresponding SFC.

inferencing tasks can be mapped contiguously. Specifically, we leverage the space-filling property to degenerate each NoI to a configuration where each curve traverses a different region of the NoI while maintaining contiguity for adjacent mapped neural layers both within and between curves.

This is illustrated in Figure 1, where each SFC consists of a head and a tail connecting a group of chiplets in a contiguous path.

Our dataflow-aware NoI is better suited to achieve higher throughput with lower energy consumption than existing NoI architectures, even in the presence of defects, for two reasons. First, the SFC structure enables a contiguous path on the NoI to exploit the dataflow. Second, the partitions created by such SFCs (revisited in Section IV) allow us to create non-overlapping regions with equal computational capabilities over the entire system. This leads to load balancing to avoid performance bottlenecks during runtime, even when employing defective chiplets onto the 2.5D system. Instead of entirely discarding chiplets with defective parts, we enable them to perform better than the state-of-the-art counterparts via a suitable hardware/software codesign strategy. By efficiently utilizing defective chiplets, we achieve reduced fabrication costs with respect to state-of-the-art alternatives considering realistic defect scenarios. The major contributions are:

1. We demonstrate how previously designed NoI topologies [11] [12] [13] [16], which represent some variants of the *k-ary n-cube* architecture, can be decomposed into an SFCed structure. We show that mapping the neural layers along the SFC-path improves robustness under defects compared to the mapping following the original topology of the NoI.
2. Extensive performance evaluations on different chiplet-based systems executing various CNN inferencing workloads show up to 63% and 76% reduction in latency and energy, respectively, compared to a state-of-the-art counterpart under defect-free and defective (to varying degrees) combinations.
3. We demonstrate the efficiency of the proposed neural layer to chiplet mapping in the presence of defects. Compared to existing NoI architectures, the proposed SFCed solution lowers fabrication cost by up to 6×.

The rest of the paper is organized as follows. Section II describes the prior work on 2.5D systems, SFCs, and NoI architectures. Section III presents the defect distribution in chiplets, while Section IV details the NoI design principles in the presence of defects. Section V presents the detailed experimental results and analysis. Finally, Section VI concludes the paper.

II. Related Work

The manufacturing cost of large monolithic chips is increasing rapidly. Fewer large chips can be implemented on a given wafer than smaller chips. Second, a defective larger die wastes more silicon area than relatively smaller dies. Most chip vendors and foundries, including TSMC, NVIDIA, Intel, and AMD, are moving towards non-monolithic alternatives such as 2.5D interposer-based systems to partition the on-chip resources into smaller discrete computing cores called *chiplets* [7]. Integrating multiple small chiplets on a large interposer offers not only significantly lower cost and higher manufacturing yield than 2D ICs, but also better thermal efficiency than 3D ICs and ease of heterogeneous integration [17]. Designing general-purpose as well as application-specific 2.5D-based systems have been explored to date. The recently proposed SIAM framework enables fast design space exploration of 2.5D-based systems [11]. The NoI paradigm becomes crucial due to the high communication demand generated by integrating many chiplets on the same substrate. Multiple NoI architectures have been proposed in the literature [10] [12]. DeFT proposes a deadlock-free routing algorithm and mitigate low reliability due to vertical-link (VL) faults on the interposer [18]. SiPterposer proposes a flexible communication fabric that supports construction of arbitrary network topologies to provide ~100% chip assembly yield at typical bonding defect rates [19]. SiPterposer employes fuses in the interposer wiring and introduces bridge chiplets across disconnected regions, focused for interposer to amortize assembly defects. HexaMesh, is a recently proposed NoI architecture, which is a projection of a 3D Mesh onto a planar graph, with up to six neighbors (hence Hexa) connected. Most of these architectures are based on conventional multi-hop interconnection architectures, such as mesh or torus. Hence, they are not scalable. A recently proposed SFC-enabled NoI, called Floret, is degenerated from a 2D-Mesh [20]. However, existing approaches assume all chiplets are fault-free. Hence, computational resources within each chiplet were identical.

SFCs [14] [15] are a specialized class of algorithmic mapping techniques that are extensively used as a locality-preserving data structure for numerous scientific applications, involving spatial and range queries. An SFC represents a linear ordering of a set of n-points in a d-dimensional space. Numerous types of SFCs have been defined over the decades, including simple schemes such as row/column major curves to more sophisticated Hilbert curves [21], Z-curve [22], or onion curves [23]. These are deterministic curves with a recursive structure to fill a space defined by an arbitrary d-dimensional point cloud. For a review of classical SFCs, please refer to [14] [15]. SFCs have found significant application in databases, parallel scientific computing, and bioinformatics; to create locality-preserving layouts for DNA nanostructures, sequence alignment, and phylogenetic inferences [24] [25] [26]. In our target application, the space we are trying to fill is provided as a 2D network topology (as shown in the examples in Figure 1). Therefore, classical SFCs do not readily apply. However, we desire the locality-preserving properties as in a classical SFC. For this reason, we define a custom SFC suited to effectively exploit the inherent dataflow pattern in CNNs and enable high-performance and energy-efficient NoI architectures.

In this paper, we bridge the gap in the existing state-of-art by proposing novel design principles for chiplet-based manycore architectures while considering the realistic defect distributions to enable high performance with energy savings. We also show how this design can be extended to multiple parent NoIs (such as Kite, 2D-Mesh, and HexaMesh).

III. Chiplet Defect Distribution

Defect distribution in chiplets: Recent research has demonstrated the potential of resistive random-access memory (ReRAM) for efficient machine learning (ML) inference [3]. ML 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. Hence, in this work we use ReRAM-based chiplet as the computational platform. Chiplet-based systems provide sufficient bandwidth and data localization to meet energy-efficiency requirements of emerging compute- and data-intensive applications [10]. As an emerging technology, chiplet-based 2.5D systems must overcome the challenges of limited yield due to defects and increased fabrication costs. We determine the fault scenarios from the well-established negative binomial distribution of the chip yield model, which was derived from the real chip manufacturing data [4] [6]. From this well-known chip defect model, we determine the best-, average, and worst-case scenarios. Yields of individual dies are necessary for estimating expected defects for a particular manufacturing process technology. The defect density D_o is initially high for a new technology. Then, it reduces significantly as the technology matures [27]. In a chiplet-based 2.5D system, each chiplet is placed onto an interposer. An interposer is essentially a large monolithic chip with more than 80% area empty [4] [6]. Within the interposer, the NoI is fabricated post-production [6]. The peripheral circuits within the ReRAM chiplets are designed with mature CMOS technology [10] [16]. Hence, the probabilities of failure both in NoI and the peripherals are much smaller compared to the ReRAM crossbars. ReRAM crossbars are used to implement the multiply-and-accumulate (MAC) operations that are the backbone of the computational kernels for the CNN inference tasks. Hence, we consider the manufacturing defects on the ReRAM crossbars of each chiplet in this work. Using the negative binomial yield model [4], we estimate the yield of each chiplet as:

$$Y_{die} = \left(1 + \mathrm{A} * \frac{\mathrm{D_o}}{\alpha}\right)^{-\alpha} \qquad (1)$$

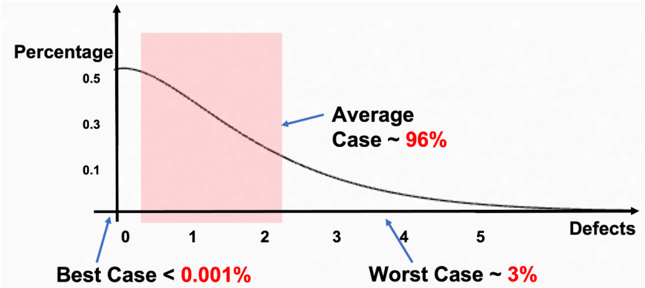


Fig. 2: Illustration of cumulative defect probabilities (Gamma distribution) for 2.5D chiplet-based system.

where α is a process-dependent clustering parameter, estimated pessimistically as 20 for a chiplet-based system, and A is the area of the chiplet [28]. We utilize the well-known Gamma distribution, as shown in (2), to determine the defect distribution within each chiplet [4] [28].

Within a local die, we assume defects follow a Poisson distribution across the entire area [27]. Multiple defects could fall into the same location within a chiplet. Hence, we model the probability $P_{defect}(d)$ that a chiplet has d defects $\forall d \in [0,1,2,\dots,t]$, where t is the total defect count:

$$P_{defect}(d) = \frac{\Gamma(d+\alpha)}{d!\,\Gamma(\alpha)} \frac{\beta^d}{(\beta+1)^{d+\alpha}} \qquad (2)$$

The probability that a chiplet has d defects is calculated using the Gamma function $\Gamma(x)$, and the constant β is defined as:

$$\beta = \frac{D_o A}{\alpha} \qquad (3)$$

***Cumulative probabilities in a 2.5D-based system*:** When considering a chiplet based system with the above-described defect-density distribution, the defect probabilities of each chiplet will vary. As a designer, it could be challenging to make a well-informed decision while looking at systems with defects since no two systems will have the same defect map for the corresponding chiplets. In this work, we consider passive interposers, i.e., no logic is implemented on the interposer. The interposer is only used for inter-chiplet communication. Moreover, the interposer has a smaller critical area than the chiplets [4]. Therefore, defects on the interposer will not affect the system performance significantly. For illustrative purposes, we consider a 36-chiplet system as an example. However, the same design principle and methodology can be extended for any number of chiplets in the presence of defects. We consider chiplets with 16 ReRAM tiles, each with 40 crossbar arrays [16]. Using this specification, we determine each chiplet area, parameter A in (3), as 2.64 mm^2 via NeuroSim. Figure 2 shows the probability of occurrences of a certain number of defects in a chiplet using (2). We propose three types of probabilistic situations to capture the generality across the whole spectrum of defects: best-case, average-case, and worst-case scenarios, as described below:

i) Best Case: In this scenario, none of the chiplets possess defects, i.e., the entire system is defect-free. As shown in Figure 2, the probability of having a defect-free chiplet using (2) is about 53%, yielding a very low probability $0.53^{36} \ll .00001\%$ that all 36 chiplets are defect-free. This unlikely scenario helps us evaluate the SFC-based NoI's performance in an the ideal (defect-free) setting.

ii) Average Case: Considering 2.64 mm^2 chiplets , we can fit 1200 chiplets on a 3-inch wafer. Among these 1200 chiplets, we pick one chiplet at random, assuming the defect densities as in (3). We continue to do the same until 36 chiplets have been chosen. Now considering 36 chiplets, we have about $\frac{1200}{36} \approx 33$ separate clusters (a 36 chiplet system is referred as a cluster). The average case considers the average of such 33 separate clusters for each of the 36 chiplets. Following (2), this results in 96% of the chiplets containing up to two defects $\forall d \in [0,1,2]$. In this situation, we have 15 defect-free and 21 chiplets with a single defect.

iii) Worst Case: This case captures the tail of the distribution to present the worst-case scenario. We consider the tail to be when there are more than 2 defects on a chiplet ($d > 2$). The worst case does not suggest all chiplets are highly defective; instead, the probability of having the most defective chiplets is the highest. Following (2), this probability turns out to be 3%. This scenario captures the other extreme of the spectrum. In this worst-case, we get chiplets with up to 5 defects, i.e., we have 20 defect-free chiplets, 9 chiplets with 1 defect, 3 chiplets with 2 defects, 3 chiplets with 3 defects, and 1 chiplet with 5 defects.

As the system size increases, the same defect distribution trend is followed, as demonstrated above using (2). For brevity, we limit further discussion and describe the NoI design and optimization approach under defects using SFCs.

IV. NoI design and optimization under Chiplet defects

In this section, we present the proposed NoI design methodology considering various CNN tasks in the presence of defective chiplets. Most previously proposed NoI architectures are degenerated from a given *k-ary n-cube* topology. Here, each point connects to *k* other chiplets along *n* directions, forming a grid-like lattice of computing cores/chiplets: i) Kite is principally derived from a torus [12]; ii) SIAM/SIMBA is indeed a 2D-Mesh [10] [11]; iii) HexaMesh [13] is a 2D projection of a 3D-Mesh with up to six neighbors; iv) SWAP [16], is an irregular NoI architecture derived from a 2D-Mesh; and v) Floret was degenerated from the initial SIAM (2D-Mesh) configuration [20].

Given the prevalence of *k-ary n-cube* type of topologies among previous works, we assume that the input topology belongs to the *k-ary n-cube* family, but also with potentially defective chiplets. The number of chiplets and their defect levels are known at the input time. The objective is to design an NoI that decomposes the input topology into a set of SFCs, each with its own sequence of chiplets to map the neural layers of any CNN task. In this work, we consider how planar topologies with defective chiplets degenerate to a corresponding SFCed topology. We denote each parent NoI topology as a planar graph *G (V, E),* which is an input to our proposed methodology that degenerates to an SFCed version from each parent.

A. SFCed NoI design

Given the need to execute various CNN inferencing tasks simultaneously, modern-day servers and high-end processors must be designed to execute multiple concurrent applications [29] [30]. We consider CNNs with different neural layer architectures – including linear (e.g., VGG), residual (e.g., ResNet), and dense (e.g., DenseNet) connections for performing inferencing tasks when designing a chiplet-based system. However, mapping different CNNs dynamically to a chiplet-based system is challenging. The common property of CNN inference tasks is that activations flow from the *i^th^* layer to the *(i+1)^th^* layer. Maintaining contiguity on the physical NoI layer between any two consecutive neural layers can reduce communication overhead. We assume each chiplet has sufficient buffers to store the intermediate activations associated with the skip connections, which flow through the same NoI links. Consistent with prior work [11] [20], each chiplet has 64KB buffer space (register files) to store the intermediate activations. We assume that all weights are transferred to the chiplets from the DRAM chiplet before performing the CNN inference tasks, consistent with prior works [11] [16].

As a defective chiplet has fewer computing resources, the neural layers may need to be spread onto multiple chiplets when there are defective chiplets, potentially increasing inter-chiplet communication. The placement of these defective chiplets must consider the system requirements, namely that there may be multiple CNN tasks that need to be dynamically mapped to the system, and each such task may consist of a different number of neural layers. Furthermore, the number of chiplets required to execute each layer could vary, often increasing due to defects.

Therefore, the underlying NoI design problem becomes one of generating multiple SFCs, each with its own sequence of chiplets to map to the neural layers of any given task. Moreover, as the different CNN tasks complete execution, the chiplets used for that task need to be released and reassigned to newer tasks (if any). If a consecutive sequence of chiplets cannot accommodate all CNN layers due to defects, the spill-over layers will need to utilize chiplets in other parts of the NoI (i.e., from other SFCs) to ensure successful completion. We provide a more formal problem definition next.

Problem formulation: Henceforth, we use the following definition of an "SFC" for our NoI design. Given a network of *n* chiplets of a certain graph topology G(V,E), with the chiplets as the nodes in V and the 1-hop links as the edges in E, an arbitrary *ψ -cover SFC* is a path of length ψ (where $\psi \leq n$) in the graph – i.e., a contiguous sequence of ψ chiplets that are connected by a series of 1-hop links along the network.

Given *G(V,E)* with *n* chiplets and a target number λ of SFCs (where $\lambda = \frac{n}{\psi}$), the problem for our NoI design becomes one of partitioning the set of *n* chiplets in *V* into λ number of SFCs such that no two SFCs share any chiplets.

In theory, the complexity class of this problem is NP-complete, as the classical Hamiltonian Path (HP) problem on a graph, which is NP-complete, reduces to our base case with $\lambda = 1$. Consequently, our solution is heuristic in nature.

While not all generic graphs contain an HP, the guarantee exists for any 4-connected planar graph, which holds for our target NoI such as a 2D-Mesh, torus, ring, hypercube, HexaMesh, and Kite (which degenerates to a 2D-Mesh if we remove the diagonal links and hence contain an HP) [31].

***Overview of NoI design**:* During degeneration of the planar *k-ary n-cube* NoI to corresponding SFC architecture (SFCed), we let the placement of the defective chiplets be guided by the expected computation patterns inherent in CNN tasks. In particular, as the first few layers of a CNN are expected to generate the largest number of activations, ensuring these high-activation layers are mapped to defect-free chiplets is crucial. With their reduced activations, the remaining layers are more amenable to getting mapped to defective chiplets. Using this general high-level idea, we propose a two-phase process to reduce latency and energy, and maximize throughput of CNN inference tasks: *(balanced assignment)* first, a defect-aware balanced assignment of chiplets to each SFC; and *(SFC placement)* next, a defect-aware placement of the SFCs on the NoI architecture so that any two chiplets consecutive on the SFC are also one-hop neighbors on the NoI. These two steps are summarized in Algorithm 1 and described next.

Defect-aware balanced assignment: We formulate the problem of determining the composition of each SFC as one of a variant of bin packing. In the classical bin packing optimization problem, the input is a set of n items $(a_0 \dots a_{n-1})$, each of a certain size $s(a_i) = (0{,}1]$, and an unlimited set of unit capacity bins. The objective is to pack (i.e., partition) the n items using the least number of bins possible [32]. This classical problem is NP-hard [34].

Our version of the problem is a constrained variant of this classical formulation. In our case, each bin is an SFC, and the items are the n chiplets such that each chiplet c has a known computational power $\mu(c)$, as defined by the defect map. Furthermore, since the number of SFCs is fixed (i.e., a positive constant λ), the number of chiplets per each SFC bin is also fixed (a positive constant $\psi = \frac{n}{\lambda}$). This implies that our version of the bin packing problem is one of generating a balanced distribution of the n chiplets across the λ SFC bins, such that each SFC would contain $\frac{n}{\lambda}$ chiplets, while the total computational power assigned to each SFC bin is approximately balanced. Ideally, the chiplets assigned to each SFC C should support a target t number of resources, where $t = \left\lceil \frac{1}{\lambda} \sum_{c\in C} \mu(c) \right\rceil$. However, this cannot be guaranteed as that would depend on the input set of chiplets and their defect rates. Consequently, we propose the following greedy heuristic approach which aims to achieve as best of a balanced assignment of chiplets to SFC bins as possible (Algorithm 1), while satisfying the SFC number and length criteria (i.e., λ SFCs each with $\psi = \frac{n}{\lambda}$ chiplets).

Our balanced assignment algorithm (Algorithm 1) first sorts the chiplets in the non-increasing order of their compute resources (line 2). Next, it assigns the sorted chiplets to bins using a simple round-robin strategy (line 4), yielding ψ chiplets per SFC. Subsequently, the algorithm swaps chiplets between the overfull bins and the underfull bins iteratively until the balancing can no longer be improved for the pair of bins under consideration (Line 5). Here, a bin is said to be "overfull" (alternatively, "underfull") if it has more than (alternatively, less than) the target t number of compute resources. Note that swapping maintains the ψ chiplets per SFC condition. To implement this shuffling of chiplets between SFCs, we use a greedy Least-Used (LU) scheme, similar to those exploited in other optimization problems with a balancing criterion [35]. More specifically, our LU scheme swaps the chiplet with the least defect from the next largest overfull bin (b_o) with the chiplet with the largest defects in the next smallest under-full bin (b_u). This is done until either the compute resources of b_o or b_u reach t, or no more swaps are possible. Figure 3 demonstrates the binning process. After

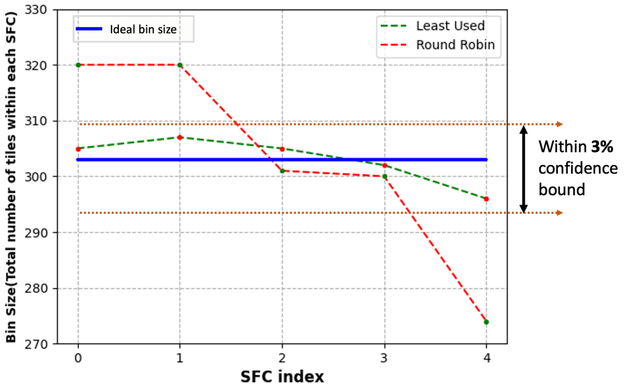


Fig. 3: Skewness reduces as the least used binning procedure allocates resources in each corresponding bins. Blue line indicates target bin size under perfect balancing.

Algorithm 1: Defect-aware SFC architecture design

```
Input: $\mathcal{C}$: a 2D grid of $n$ chiplets represented as a graph $G(V,E)$
    $\lambda$: the desired number of SFCs
    $\mathcal{D}$: Defect map of each chiplet with compute capacity $\mu(c)$
Output: A list of $\lambda$ SFCs: $\Pi = \{\Pi_0, \Pi_1, \dots \Pi_{\lambda-1}\}$, where each
    SFC $\Pi_i$ is of size $\psi$; $\psi = \lceil \frac{n}{\lambda} \rceil$
 1: $t \leftarrow \lceil \sum_{c\in\mathcal{C}} \frac{\mu(c)}{\lambda} \rceil$ /* Target resource for SFC */
 2: $C_{sort} \leftarrow$ Sort $n$ chiplets in non-increasing order
 3: Initialize $\lambda$ bins of size $\psi$; $\mathcal{B} = \{b_0, b_1, \dots b_{\lambda-1}\}$
    /* Step1: Creation of bins using least used */
 4: Assign $C_{sort}[i]$ to SFC $\Pi_{i \bmod \lambda}$
 5: Exchange elements until $\sim$ t $\forall b \in \mathcal{B}$
    /* Move elements to underfull bins until target $t$ is achieved $\forall b \in B$ */
 6: $[\langle H,T \rangle] \leftarrow$ Assign a list of $\lambda$ $\langle$head, tail$\rangle$ chiplet position pairs
    in $\mathcal{C}$
    /* Step2: SFC creation using bins $\mathcal{B}$ */
 7: for all $\langle h_i, t_i \rangle \in [\langle H,T \rangle]$ do
 8:     Initialize $\psi \leftarrow \lceil \frac{n}{\lambda} \rceil$
        /* i.e., TSP tour length for each SFC */
 9:     Initialize $\Pi_i$ to an empty array of (TSP tour) size $\psi$
10:     $\Pi_i \leftarrow ComputeTSP(G(V,E), \langle h_i, t_i \rangle, \psi)$
11:     Update graph $G$ by removing all edges incident on $\Pi_i$
12: end for
13: $\Pi \leftarrow \bigcup_i \Pi_i$
14: return $\Pi$
```

Algorithm 2: Defect-aware mapping algorithm

```
Input: Workload with multiple CNNs ($\mathcal{W} = \{w_i\}$) each with
    multiple layers
    $\mathcal{C}$: the set of $n$ chiplets, where each chiplet $c \in \mathcal{C}$ has compute
    capacity $\mu(c)$
    $\Pi$ : a set of $\lambda$ SFCs $\{\Pi_0, \Pi_1, \dots \Pi_{\lambda-1}\}$, with each SFC
    internally ordered in non-increasing order of chiplet defect
    rates from their respective heads $h$
Output: Mapping of each CNN $w \in \mathcal{W}$ to a distinct subset of
    chiplets (i.e., $\Phi : \mathcal{W} \rightarrow 2^{\mathcal{C}}$ such that the mappings of chiplets
    are mutually disjoint)
 1: Init. $next = 0$ /* allocation to start at the first
    chiplet $\Pi(1)$ */
 2: Init. $n' = n$ /* the running count of the number of
    available chiplets */
 3: Init. a countdown timer at each SFC head to 0
    (i.e., $h.timer = 0, \forall h \in H$)
 4: for all $w \in \mathcal{W}$ do
 5:     $\Theta \leftarrow$ number of compute resources required by $w$
        (rounded to the next integer)
 6:     Compute $\tau_{jump}(h,w) = h.timer + \tau_{exec}(h,w)$
        for each $h \in H$
 7:     $h_{best} \leftarrow \arg\min_h (\tau_{jump}(h,w))$
 8:     if $(\tau_{jump}(h_{best},w) < \tau_{exec}(next,w))$ then
 9:         $next = pos(h_{best})$ /* Jump over the defective
            chiplets to the next available head */
10:     else
11:         continue /* Mapping position $next$ does not get
            updated */
12:     end if
13:     $\langle \Phi(w), n' \rangle = \mathtt{mapCNN}(w, \Pi, next, \Theta, n')$ /* Map $w$ to a
        sequence of $\Theta$ compute resources starting at
        $next$ position; return the updated $n'$ */
14:     Update $next \leftarrow (\Phi(w).lastindex + 1) \bmod n$
15: end for
16: return $\Phi$
```

the chiplets are assigned to SFC bins, the chiplets of each SFC are internally sorted in non-increasing order of their computational resources (line 6).

Defect-aware SFC placement: Next, we place each SFC onto the NoI architecture to guarantee NoI contiguity between consecutive chiplets of any SFC. To achieve this, we first choose 2λ locations on the NoI that can house the λ head-tail chiplet pairs of the SFCs. Here, we note that reducing the average number of hops separating the tail of an SFC from the head of another SFC is important during the subsequent CNN layer mapping phase, since the same CNN task may possibly use chiplets from two or more SFCs. Therefore, our search objective becomes minimizing the average path length d between the tail of one SFC to the heads of other non-overlapping SFCs:

$$Minimize\colon\ d = \frac{1}{p} \sum_{i,j\in[0,\lambda-1]} \left|t_i - h_j\right|_{where\ i\neq j,\ p=2\binom{\lambda}{2}} \quad (4)$$

The distance between any tail-to-head pair is calculated as the Manhattan distance over the 2D grid. Minimizing this average distance measure d is imperative as communication delay increases when data flows between the tail of one SFC to the head of another SFC. The above principle determines the placement of each chiplet within each SFC.

Once all SFC bins are created with the defective chiplets, and the head-tail pairs are selected, the chiplet set assigned to each SFC bin is linearly ordered. This task is one of using the NoI topology's one-hop links and neighborhood information so that a contiguous path with ψ consecutive chiplets, connected via 1-hop links, is detected between the head and tail chiplets (which act as the two sentinel ends of the path). The problem is formulated as a traveling salesman problem (TSP) over the $(\psi - 2)$ chiplets on a planar graph [33]. From each chiplet (head in this case), we visit other chiplets (nodes), creating a Hamiltonian path. Each visited chiplet is stitched as part of the SFC, while we keep on choosing a chiplet from the head of the respective bin and appending them onto the SFC. By placing the chiplets in a non-increasing order of defects, we ensure that the highest activation layers can be mapped to least defective chiplets.

B. *Neural network to Chiplet Mapping under Defects*

Next, we describe the defect-aware mapping of CNN inferencing tasks to the SFCed NoI architecture. Recall that targeted workflow is a concurrent list of CNN inferencing tasks. For each CNN task, we map the task to the current chiplet location within an SFC or jump to any available head. Let Π denote the set of all λ SFCs. Algorithm 2 details the major steps of this mapping procedure. Let the workload W be a queue of multiple CNN tasks. For each $w \in W$, we first compute the required number of computing resources. Initially, all chiplets across all λ SFCs of Π are considered available. We keep a countdown timer $h.timer$ for each of the λ heads that is initialized to 0. We track a $next$ pointer to point to the next chiplet along Π that is due for assignment. The $next$ pointer is where the mapping starts. Initially, $next$ is initialized as the head chiplet of the first SFC (Π_0). In essence, the algorithm updates this *next* location to map the CNN to the chiplet, which minimizes the execution time and then maps the networks.

More formally, the algorithm must choose between one of two possible mapping locations: either the next chiplet in order of the current SFC, or the head of a different SFC. This decision is based on which mapping choice would result in the lower execution time, as highlighted in line 8 of Algorithm 2. With the CNN layer to chiplet mapping, we keep track of the expected countdown time of each head. This time is derived from the communication traffic arising from the inter-layer activations of the CNN mapped on that head. The major function that computes $\Phi(w)$ for any given task w is $MapCNN(w, \Pi, next, \Theta, n)$, shown in line 13 of Algorithm 2. This function maps the task w to a sequence of Θ computing resources, starting from the $next$ position along Π. The actual chip let coordinates for this next position is given by $\Pi^{-1}(next)$. The $MapCNN$ function returns when all layers of w have been successfully mapped. Note that it is possible that along the mapping process, the next chiplet to be assigned is occupied with another CNN task. In this case, the procedure waits until another head or $next$ chiplet becomes available depending on the condition shown in Line 8 in Algorithm 2. No deadlocks are possible since we test all available heads, and the timers associated with chiplets are countdown timers.

V. Experimental Results

In this section, we present a thorough performance evaluation of the SFCed architectures in the presence of chiplet defects considering concurrent CNN inferencing tasks. We also compare the proposed SFCed NoI with respect to corresponding parent NoIs.

A. *Experimental Setup*

i) System specification and evaluation setup: To demonstrate the efficacy and scalability of the proposed SFCed architectures as a function of chiplet defects, we consider three different system sizes (*n*) with 36, 64, and 100 chiplets. Each chiplet has $2.64\ mm^2$ area, with 16 tiles, each tile consisting of 40 PEs, and each PE consisting of 128x128 crossbar arrays [11] [16]. We use a modified NeuroSim to partition and map the concurrent CNN tasks onto a 2.5D-based system [34]. The inter-chiplet traffic is generated by the activations between the neural layers. The activations generated by residual skip connections are also part of the communication traffic. All the NoI topologies are simulated using the BookSim simulator [35]. The inputs to the BookSim simulator are the connectivity between NoI routers and the inter-chiplet traffic generated for each CNN inference task within the workload. We determine the area, latency, and energy consumption of the NoI using a modified version of BookSim to capture custom NoI topologies considered in this work. We use the Nvidia ground-referenced signaling (GRS) parameters for chiplets on a 32nm technology to evaluate the NoI area and power consumption [10]. Table I shows the other system-level parameters considered in the performance evaluation [11] [17]. In our work, each chiplet consists of 16 tiles following previous work [16] [11]. It should be noted that as the chiplet size decreases, the number of defects per chiplet could potentially reduce. However, the computational and storage capabilities of each chiplet also reduce. Hence, weights of each neural layer and its corresponding activations are spread onto multiple chiplets, giving rise to higher inter-chiplet traffic. This in turn leads to overall system level latency overhead and lower energy efficiency. We note that the experimental analysis and performance evaluation

TABLE I: NOI 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**

considered in this paper can extend to other technology parameters.

ii) Datasets and CNN workloads: We evaluate the performance and energy efficiency of the original NoIs with respect to their SFCed architecture in the presence of defects on multiple concurrent CNN inferencing tasks. Table II shows different neural networks executed on the corresponding datasets and their number of parameters. As the system size increases, we use ImageNet-based CNNs with more parameters to illustrate the merits of the proposed architecture. Table III shows the naming convention of the CNN tasks in each workload (WL) along with their total number of parameters running on (a) CIFAR-100 and (b) ImageNet datasets. Tables III(a) & (b) show the CNN tasks queue to be executed on the 2.5D system. Various combinations of the neural networks in Table II are executed concurrently to capture the workloads (WL) considered in the experimental setup. We evaluate the 36-chiplet system using CIFAR-100 dataset. For scalability, we evaluate 64 and 100 chiplet system on ImageNet-based workloads as the number of parameters is in the order of billions. As an example, WL1 consists of sixteen instances of NN_2 (ResNet34), along with one instance of NN_7 (VGG19), and so on. We cover the whole spectrum by randomly choosing any combination of the CNN tasks. Note that the general concept behind our NoI design applies to any deep learning inference tasks. Additionally, the workloads consist of multiple ResNet and DenseNet models to validate the advantages of SFCed NoIs with skip connection-based CNNs.

TABLE II: LIST OF NEURAL NETWORKS FOR INFERENCING ALONG WITH THEIR CORRESPONDING NUMBER OF CNN PARAMETERS WITH (A) CIFAR-100, (B) IMAGENET DATASET

(a)

Name	Neural Network	Number of Parameters (CIFAR100)
NN_1	ResNet18	1.8M
NN_2	ResNet34	2.79M
NN_3	ResNet50	4.15M
NN_4	ResNet110	9.42M
NN_5	ResNet152	12.96M
NN_6	VGG16	1.67M
NN_7	VGG19	1.91M
NN_8	DenseNet40	1.6M

(b)

Name	Neural Network	Number of Parameters (ImageNet)
NN_9	ResNet18	24.76M
NN_{10}	ResNet34	36.5M
NN_{11}	ResNet50	25.94M
NN_{12}	ResNet101	9.42M
NN_{13}	ResNet110	43.6M
NN_{14}	ResNet152	54.84M
NN_{15}	VGG19	93.4M
NN_{16}	DenseNet169	892.72M

TABLE III: LIST OF CNN TASKS IN A WORKLOAD FOR INFERENCING ALONG WITH THEIR TOTAL NUMBER OF PARAMETERS WITH (A) CIFAR-100, (B) IMAGENET DATASET

(a) – CIFAR100

Name	List of CNNs in a workload	Total number of parameters
$WL1$	$16NN_2, NN_7, 5NN_3, 3NN_8, NN_5, NN_7, 4NN_4, NN_6, NN_1, NN_3, NN_6$	133M
$WL2$	$NN_7, NN_1, NN_6, NN_1, 11NN_3, 3NN_3$, NN$_7$, $3NN_4$	88M
$WL3$	$NN_7, NN_6, NN_5, 3NN_4, 9NN_8, 4NN_2, 12NN_1, 5NN_3, NN_6$	114M
$WL4$	$NN_5, NN_7, NN_1, NN_3, NN_6, 5NN_3, 3NN_8, 16NN_2, 4NN_4, NN_6, NN_7$	133M
$WL5$	NN_5, NN_8, NN_1, NN$_3$, $NN_4, 6NN_2, 4NN_6$, $11NN_4, 5NN_5, 2NN_6$	240M

(b) - ImageNet

Name	List of CNNs in a workload	Total number of parameters
$WL6$	$16NN_{10}, NN_{15}, 5NN_{11}, 3NN_{16}, NN_{13}, NN_{15}, 4NN_{12}, NN_{14}, NN_9, NN_{11}, NN_{14}$	1.1B
$WL7$	$2NN_{15}, NN_{14}, NN_{13}, NN_{12}, 7NN_{11}, 2NN_{12}, NN_{16}, NN_{12}$	1.4B
$WL8$	$NN_{15}, NN_{14}, NN_{13}, 3NN_{12}, 9NN_{16}, 4NN_{10}, 12NN_9, 5NN_{11}, NN_{14}$	8.8B
$WL9$	$NN_{13}, NN_{15}, NN_9, NN_{11}, NN_{14}, 5NN_{11}, 3NN_{16}$, $16NN_{10}, 4NN_{12}, NN_{14}, NN_{15}$	3.8B
$WL10$	$NN_{13}, NN_{16}, NN_9, NN_{11}, NN_{12}, 6NN_{10}, 4NN_{14}, 11NN_{12}, 5NN_{13}, 2NN_{14}$	1.8B

TABLE IV: NUMBER OF SFCS IN EACH DEGENERATED ARCHITECTURE

System Size	Kite	HexaMesh	SWAP	SIAM	Floret
36	6	4	3	6	6
64	4	6	3	4	4
100	4	5	4	5	5

Due to chiplet defects, a given neural layer may need to be distributed to multiple chiplets, giving rise to an increase in inter-chiplet traffic. This effect is particularly pronounced in layers where the number of activations is significantly higher than the rest. In the initial layers of CNNs, convolutional layers extract basic features (such as edges, textures, and colors) while preserving spatial resolution. The convolutional filters applied to the input image at this stage do not reduce the spatial dimensions. Hence, the activation from the initial layers remains large. Higher activations in the starting layers give rise to more data exchange among communicating chiplets. Chiplet defects, due to the unavailability of compute tiles, leads to higher volume of inter-chiplet traffic. This, in turn, increases latency and reduces energy efficiency.

iii) Baseline NoI design: We compare the performance of the SFCed NoIs in the presence of defects against four baselines: Kite, HexaMesh, SIAM, and application-specific NoI architecture SWAP [12] [11] [16]. Kite is primarily a Torus-based NoI, and SIAM is a 2D-Mesh NoI. HexaMesh is a concentrated mesh NoI with staggered chiplet placements with up to six router ports for any internal chiplet, principally a projection of a 3D-Mesh onto a planar structure. The application-specific SWAP NoI is an irregular architecture where the chiplets and the associated links are placed per specific design time considerations for a given set of CNN applications. We set the same system parameters and evaluate over the same CNN workloads for all architectures (Kite, HexaMesh, SIAM, SWAP, and their SFCed counterparts) for a fair comparison. Kite, HexaMesh, SIAM, and SWAP are the original/parent NoIs referred to as '*Case-I*,' and their

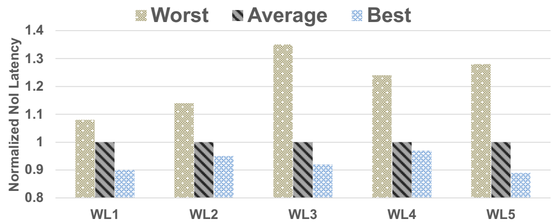


Fig. 4: Latency comparison between worst-, average-, and best-case defect scenarios for 36 chiplet system for SFCed Kite.

SFCed counterparts are referred to as '*Case-II*': i.e., Case-I corresponds to the Parent NoI; and Case-II to the corresponding SFCed NoI. We map each CNN layer in Case-I using a nearest neighbor mapping algorithm that allocates each incoming CNN layer to the closest available chiplet to ensure minimum communication delay among communicating cores. This ensures maximum achievable performance as communicating chiplets are mapped as close as possible on the NoI. However, these three architectures have multi-hop paths between chiplets, which are more pronounced under chiplets with defects as activations from one neural layer may spread over multiple chiplets. Hence, having contiguous available chiplets may not be possible as the number of CNNs increases when the interconnection topology is inherently multi-stage (mesh- or torus-based). This could lead to consecutive neural layers being mapped to far-apart chiplets through multi-hop paths. For bigger system sizes, the number of multi-hop paths increases. On the contrary, the SFCed group of NoI architectures always ensures that communicating CNN layers are mapped to contiguous chiplets, and we start mapping on a defect-free head. Hence, SFCed NoIs (Case-II) consistently outperform their (state-of-the-art) parent NoI architectures with higher energy efficiency.

B. *NoI Performance and Energy Analysis*

Determining the optimum number of SFCs λ for each considered system size is the first essential step. The optimum number of SFCs depends on the distance between the tail of one SFC to the heads of the other non-overlapping SFCs (refer eq. (4)). We consider an iso-chiplet area, i.e., individual chiplet size remains constant as the system size scales. Hence, the number of SFCs λ remains within a limited range for varying system sizes. For example, the number of optimum SFCs for SFCed Kite for 36-, 64-, and 100-chiplet systems are 6, 4, and 4, respectively. Table IV shows the number of corresponding SFCs in each SFCed NoI. It should be noted that Floret is a SFC architecture that was degenerated from 2D-Mesh (SIAM). Hence, Floret and SIAM have the same number of SFCs.

Second, the proposed SFCed architecture has an inherent advantage over parent NoI architectures in terms of power consumption due to smaller routers and fewer links in SFCs.

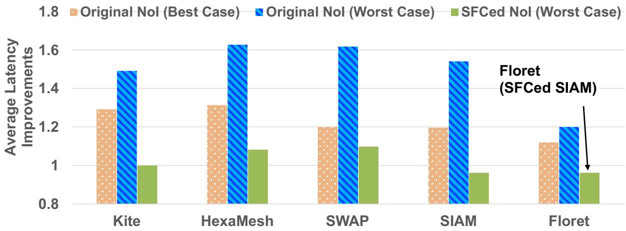


Fig. 5: Average latency improvement(normalized) for 36 chiplet system comparing defect-free and worst-case Case-I with respect to worst-case Case-II.

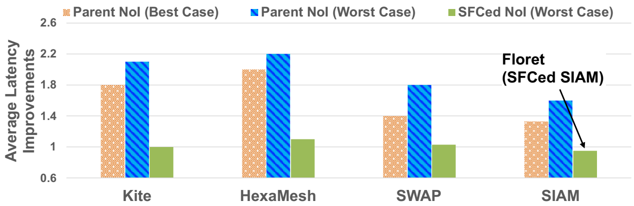


Fig. 6: Average latency improvement (normalized) for 64 chiplet system comparing defect-free and worst-case Case-I with respect to worst-case Case-II.

Except for chiplets designated as heads and tails (h, t), the $(\psi - 2)$ chiplet on each SFC has only two associated router ports. Only the heads and tails in the system will have more than two ports based on the inter-SFC connectivity. We observe that six-port and four-port routers are most frequent in HexaMesh and Kite [12]. SIAM with 2D-Mesh NoI consists mostly of routers with three and four ports. Moreover, SWAP primarily uses three-port routers [16]. Smaller routers in the SFCed architecture also correspond to fewer links. It should be noted that reducing the number of links and router ports alone does not necessarily lead to performance and energy efficiency. To achieve these benefits, it is crucial to consider iso-computation across the spatial region of the 2.5D system. The communication volume directly depends on the neural layer to chiplet mapping. If a neural layer is partitioned across multiple chiplets, the traffic volume on the NoI increases. This leads to performance degradation and higher energy consumption. Therefore, computational uniformity should be maintained for high energy efficiency. In the presence of defects, the Case-I NoI architectures have nonuniform computational capabilities due to lack of iso-partitions. HexaMesh and Kite, for example, have mostly two-hop links, and the routers are inherently larger. SIAM, principally a 2D-Mesh, has single-hop link connections to its neighboring chiplets. However, SIAM has larger routers with a higher number of router ports. SWAP has fewer links and smaller router ports, but not all links are necessarily single hop. Within each λ SFCs, each partition has equivalent computational capabilities. Moreover, all the redundant links are omitted, and all the SFC routers are small. These factors improve NoI performance and energy efficiency. Moreover, smaller routers, fewer links, and reduced NoI area, hence the fabrication costs and carbon footprint, are highlighted in the next section.

We benchmark Case-I's latency and energy consumption compared to Case-II for five different CNN workloads on each dataset (WL1-WL5 on CIFAR-100; WL6-WL10 on ImageNet) for all the system sizes. We consider three defect scenarios (best-, average-, and worst-case described in Section III). It should be noted that the best-case principally corresponds to an ideal, defect-free situation. Each bar plot is normalized with respect to SFCed Kite in the worst-case defect configuration.

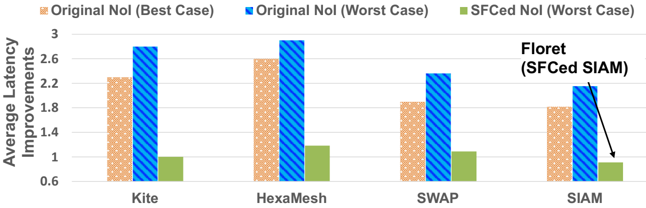


Fig. 7: Average Latency improvement (normalized) for 100 chiplet system comparing defect-free and worst-case Case-I with respect to worst-case Case-II.

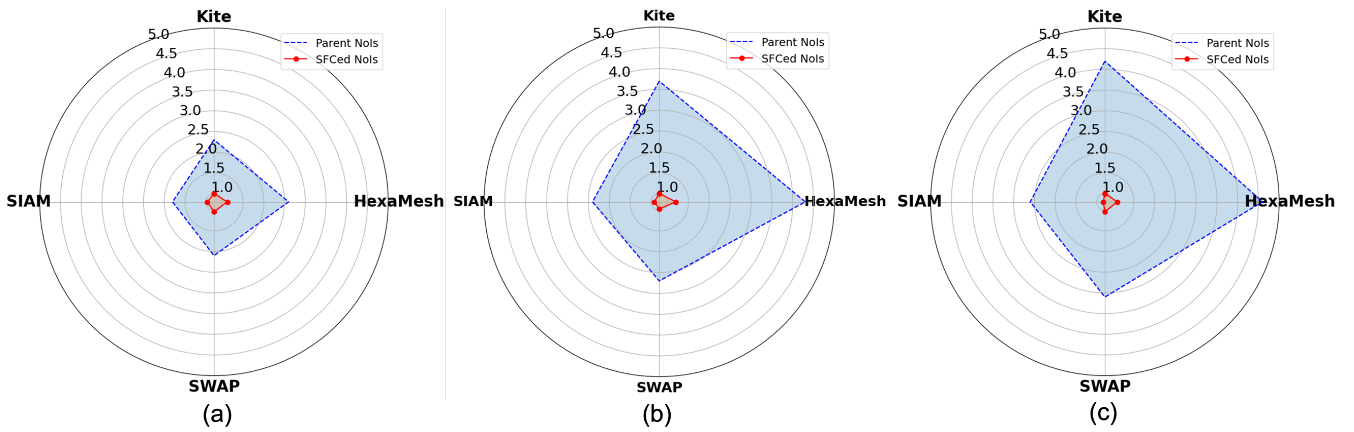


Fig. 8: Radar plots showing the energy efficiency of SFCed NoIs under worst-case defects over their parent NoIs under best-case defects for (a) 36 chiplets; (b) 64 chiplets; (c) 100 chiplets system. SFCed NoIs have smaller bounded area and hence higher energy efficiency in comparison to their parent counterparts. As the system size scales, the ratio of the enclosed area between SFCed and parent NoIs increases; hence the energy efficiency scales with bigger system size.

Latency: We first evaluate the latency degradation from the best-case to the worst-case defect scenario. For this, we consider only the SFCed Kite architecture as an example. Figure 4 shows the latency comparison among the best-, average-, and worst-case of defects considering WL1-5 on a 36-chiplet system. As mentioned earlier, neural layers are spread over multiple chiplets due to defects, increasing the inter-chiplet traffic. This gives rise to performance penalty in a more defective system running the same workload. On average, the worst-case scenario incurs a 19% latency penalty with respect to the best-case counterpart. The highest latency penalty is 36% for the worst-case. The merits of the SFC architecture are best understood when its performance under the worst-case defect scenario is compared to other NoI counterparts under the best-case scenario (i.e., having no defective chiplets).

Figure 5 shows the average latency improvement of each NoI for the 36-chiplet system considering workloads WL1 to WL5 in various degree of defects. SFCed architectures outperform their parent baselines for all defect configurations, including the corresponding best-case scenario. For example, SFCed Kite improves the latency on average by ~26%, and ~44% compared to best-case and worst-case Kite architecture, respectively. A similar trend is noted across all degenerated NoIs with respect to their original counterparts. Similarly, as the system size scales, the performance improvement for SFCed NoI is more significant since collocating communicating neural layers to neighboring chiplets becomes easier during the mapping along the SFC path. It should be noted that SFCed Floret is the same as SFCed SIAM and achieves the highest performance among all SFCed counterparts. This is because Floret was degenerated from a 2D-Mesh topology. Under the worst-case defect scenario, the set of inter-chiplet links, the placement of the defective chiplets into SFCs, and the defect-aware mapping strategy – all of them collectively yield the same NoI architecture as SIAM (2D-Mesh). For this reason, henceforth we show only results for SFCed SIAM instead of showing both Floret and SIAM in our further investigations.

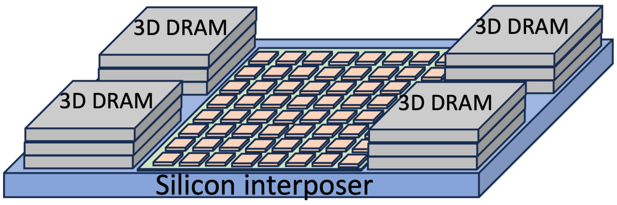


Fig. 9: A 64 chiplet 2.5D system with four 3D-DRAM stacks.

Figures 6 and 7 show the latency of each NoI architecture for the 64-and 100-chiplet systems, respectively. SFCed performs 46%, 40%, and 42% better than Kite, SIAM, and SWAP, respectively, considering the worst-case defect scenario for the 64-chiplet system. As the system size scales up further, the effect of defects is more pronounced. For bigger systems, inter-chiplet traffic due to defects increases. This leads to more multi-hop paths for Case-II due to the unavailability of contiguous chiplets to map to. The highest latency improvement of 65% for Case-I is achieved for WL7, considering the worst-case defect scenario on the 100-chiplet system with respect to Kite and its SFCed counterpart, as shown in Figure 7. In the average defect scenario, the performance improvement for SFC is always more than at least 25% compared to all the other NoI topologies for all the system sizes considered here.

Energy: SFCed NoIs achieve significant energy savings besides reducing the inference latency. They avoid multi-hop communication traffic, boosting energy efficiency. Figures 8(a)-(c) show the radar plots capturing the achievable energy improvement for 36-, 64- and 100 chiplet systems, comparing the worst-case of defects for SFCed NoI to best-case of defects for their parent counterparts in defect-free configuration. SFC always increases the energy efficiency for all system sizes and defect scenarios. The highest energy savings are achieved for the 100-chiplet system considering the workload WL9, where SFCed NoIs reduce the energy consumption with respect to Kite, SIAM, and SWAP by 78%, 62%, and 65%, respectively.

Notably, SFCed NoIs under worst-case defects outperform all other topologies with no defects. The highest improvement for SFC is observed in comparison with Kite. Even with the worst-case defects, SFCed Kite achieves on average 63% and 78% reductions in latency and energy, respectively, compared to Kite with no defects. This analysis shows that the SFCed architectures outperform all the parent counterparts. This points towards the merits of the SFCed NoI topologies. Adopting SFCed NoI not only ensures high performance but also energy efficiency and the highest robustness to manufacturing defects. Next, we illustrate the fabrication cost benefits of the SFCed Architectures.

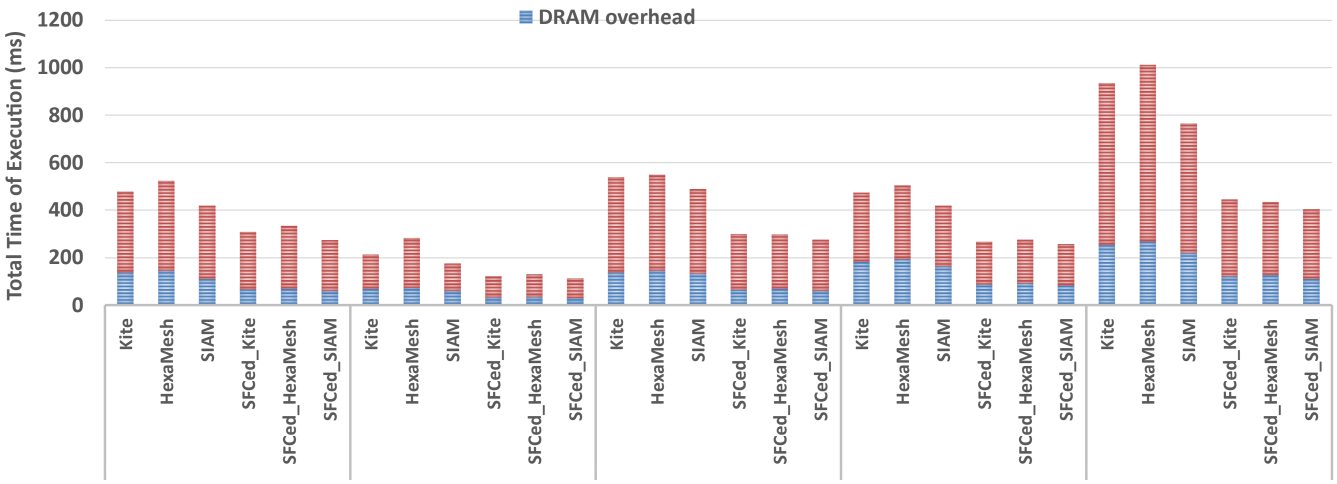


Fig. 10: Overall execution time considering DRAM overhead for a 36 chiplet system under average case of defects.

C. DRAM overhead for NoI Architectures

In this subsection, we quantify the impact of DRAM loading on the overall execution time. The targeted architecture is expected to have sufficient memory capacity to store the weights and activations of various well-known DNN workloads. Considering the storage capacity of each chiplet with 8-bit precision is 2.5MB, a 100-chiplet system has a total storage capacity of ~250 MB. For the smallest system size of 36 chiplets, the storage capacity is ~90 MB. This storage is enough, considering the largest CNN tasks (VGG19 on ImageNet Dataset) require 137 MB at the same 8-bit precision. Similarly, for the smallest considered 36-chiplet system, the storage capacity is enough for the biggest neural network on CIFAR-100. However, the memory requirement increases when we execute multiple concurrent CNN tasks. Hence, we need to read/write weights and activations from/to an external memory for the corresponding CNN task. The DRAM loading depends on the availability of the weights on the chiplet system. Hence, DRAM loading will not occur for each CNN task and depends on the order of occurrences, which is a function of the workload. As a result, there will be a variable additional latency penalty added to the overall execution time corresponding to that. While evaluating the DRAM access latency, we consider that the DRAM is connected through the interposer with the chiplet system. Figure 9 shows the 2.5D-chiplet system with four 3D stacks of DRAM on the interposer, consistent with existing works [6] [12]. We use RAMULATOR for estimating DRAM access times considering standard state machine referenced here [36] [37]. Figure 10 shows the overall execution time when executing workloads WL1-WL5 on 36 chiplet system under average case of defects. These workloads represent a class of concurrent CNN tasks. Depending on the specific workload, the DRAM access varies from 21% to 34.8% of the overall execution time. The variation in overall DRAM loading time is due to the workload characteristics considered here. For example, in WL5, as there is less immediate repeatability of CNNs, the DRAM loading overhead is higher than WL1. In the case of WL1, multiple instances of ResNets34 are executed, reducing the need for DRAM loading after each CNN. However, this relatively high DRAM access time arises due to random ordering of concurrent CNN tasks that creates very high degree of variability among CNNs being executed. This represents a very pessimistic situation. In reality, we expect an inference task to be executed multiple times using the same set of CNNs. In that case, the DRAM access time will be amortized over multiple inference passes with same CNN tasks. As an example, if 12 ResNet34 CNNs are executed concurrently for 20 passes on a 36-chiplet SFCed_HexaMesh, the overall DRAM overhead will only be 1.5% of the total execution time. It should be noted that the overall advantage of the SFCed architectures remains unchanged with respect to *k-ary n-cube* counterparts even in presence of extremely variable concurrent CNN tasks. Hence, though DRAM execution incurs additional performance overhead based on the concurrent CNN tasks being executed, this does not impact the advantages of the SFCed NoI architectures demonstrated in this work.

D. Fabrication Costs of SFCed Architectures

Over the last two decades, we have witnessed a dramatic rise in computing demand fueled by new applications at the edge and cloud scale. Higher computational requirements lead to higher costs for manufacturing and maintaining such systems. This section discusses the relative fabrication cost reductions if the defective chiplets were discarded instead of utilizing them. For example, considering the 36-chiplet system for both the average-and worst-case, following (2), we know that the system will have a certain number of defective chiplets. The normalized fabrication cost of chiplets is expressed as [11]:

$$C_{system} = \frac{L_{ref}}{L} \times e^{-D_0(A_{ref} - A_{system})} \qquad (5)$$

where L_{ref} is the number of chiplets per wafer in the reference system, and L is the number of chiplets per wafer for the system under consideration. The parameter D_0 represents the defect density, and A_{ref} is the NoI area of the reference system. We consider a 2.5D system designed by AMD with 864 mm^2 area as the reference in this work [6]. The system area consists of the chiplets and the NoI. Using (5), we can compare the fabrication cost of two different chiplet systems. For example, the fabrication cost for a defective 2.5D system is:

$$C_{Defective} = \frac{L_{ref}}{L} \times e^{-D_0(A_{ref} - A_{Defective})} \qquad (6)$$

Similarly, the fabrication cost of the defect-free 2.5D system:

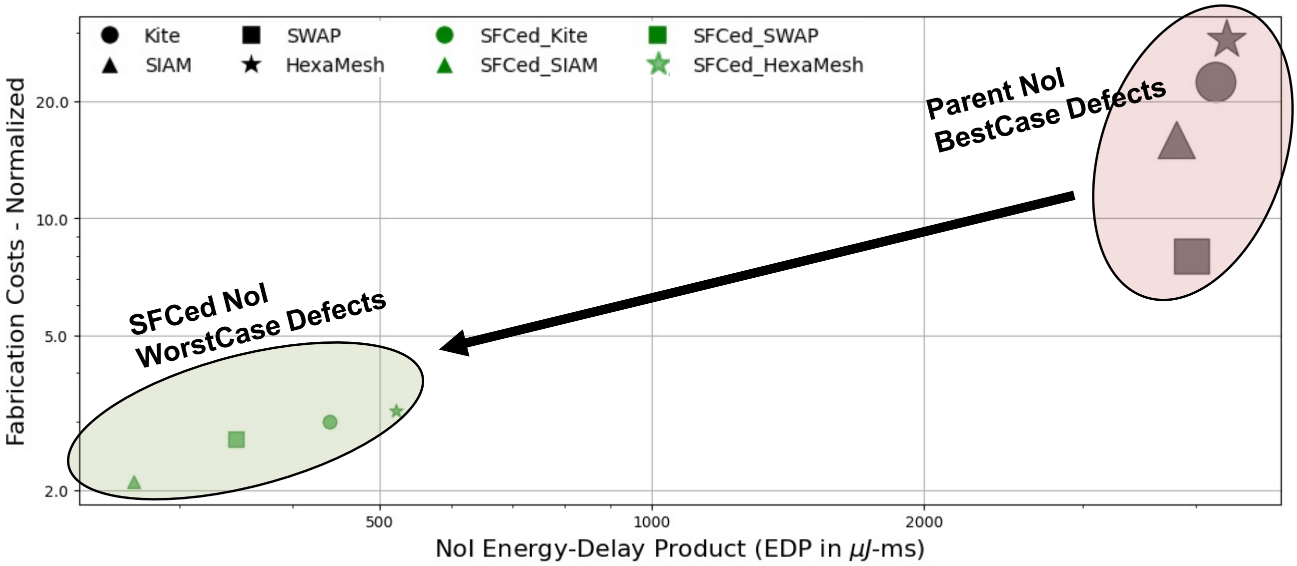


Fig. 11: Energy-Latency reduction trend (log-scale) of SFCed NoIs (worst-case defects) compared to their parent counterparts (Kite, SIAM, SWAP, HexaMesh) in defect-free systems) running workload WL6-10 on ImageNet dataset.

$$C_{DefectFree} = \frac{L_{ref}}{L} \times e^{-D_0(A_{ref}-A_{DefectFree})} \quad (7)$$

where $A_{Defective}$ and $A_{DefectFree}$ correspond to the total area of defective and defect-free systems. Therefore, the ratio of the fabrication costs can be expressed as:

$$\frac{C_{Defective}}{C_{DefectFree}} = e^{-D_0(A_{DefectFree}-A_{Defective})} \quad (8)$$

Considering the average- and worst-case scenarios discussed in Section III, the relative fabrication cost primarily boils down to the additional area of the discarded chiplets and the NoI. The NoI area depends on the size of the routers, number, and length of inter-chiplet links. Hence, designing an optimized NoI, which considers the defect distribution in chiplets is necessary for creating 2.5D systems.

Figure 11 plots the overall trend for comparing NoI architectures (SFCed NoI with worst-case defects vs Parent NoI with defect free system) for 100 chiplet system. The y-axis is normalized with respect to a 36 chiplet SFCed Kite. SFCed NoIs have lower energy-delay product (EDP) and reduced fabrication costs than parent NoIs. The marker size is proportional to the corresponding NoI area. As an example, there is a 10x EDP reduction with 6x fabrication cost improvement for SFCed HexaMesh due to reduced router sizes and robustness to defects. Hence, the SFCed NoIs are towards the left compared to parent NoIs. This shows a significant improvement in the performance, energy, and fabrication costs for our SFCed architectures compared to the parent NoIs. It is observed that the bottom-leftmost point (with minimum fabrication cost and EDP) is SFCed SIAM, which is essentially Floret. This is because all the links in SFCed SIAM (Floret) are single-hop connections, whereas the link length varies depending on the parent NoI for Kite, HexaMesh and SWAP, respectively. This illustrates the importance of choosing the right parent architecture when designing SFCed counterparts for CNN inference tasks. Overall, our results suggest the high suitability of SFC-based architectures for CNN inference tasks compared to more traditional designs.

VI. Conclusion

The emergence of 2.5D chiplet platforms provides a new avenue for compact scale-out implementations of emerging compute- and data-intensive applications. However, progress in designing chiplet-based systems is impeded by silicon defects. This paper presents a space-filling curve (SFC)-based NoI architecture that achieves high performance and energy efficiency even in the presence of silicon defects. Instead of discarding defective chiplets, SFCed NoI optimizes their utilization. SFCed NoIs enable a dataflow aware NoI to design 2.5D architecture that outperforms their parent *k-ary n-cube* based NoIs. Comprehensive experimental evaluation with different system sizes and diverse CNN inferencing workloads demonstrate that SFCed NoIs on average achieve up to 2.3x and 3.5x reduction in latency and energy consumption. SFCed NoIs enable robustness to defects in comparison to a situation where the defective chiplets were discarded and hence reduce their NoI area and hence the fabrication cost.

References

[1] W. Liu et al., "A survey of deep neural network architectures and their applications," *Neurocomputing,* no. 234, 2017.

[2] Z. Wu et al., "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems,* vol. 32, no. 1, 2020.

[3] J. A. Carballo et al., "ITRS2.0:Towardare-framingofthesemiconductor technology roadmap.," in *IEEE 32nd Intl. Conf. Computer Design*, Oct 2014.

[4] D. Stow et al., Cost-Effective Design of Scalable High-Performance Systems Using Active and Passive Interposers, IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2017.

[5] *International technology roadmap for semiconductors 2.0, 2015 edition, system integration. Report Ch 1, 2015.,* Semiconductor Industry Association, 2015.

[6] A. Kannan, N. Jerger and G. Loh, "Enabling interposer-based disintegration of multi-core processors," in *Proceedings of the 48th International Symposium on Microarchitecture (MICRO-48)*, New York, 2015.

[7] N. Jerger, A. Kannan, Z. Li and G. Loh., "NoC Architectures for Silicon Interposer Systems: Why Pay for more Wires when you Can Get them (from your interposer) for Free?," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, USA, 2014.

[8] *https://www.cloudera.com/content/dam/www/marketing/resources/ebooks/how-to-take-ai-applications-from-concept-to-reality-with-cml-on-aws.pdf.landing.html.*

[9] S. Bergsma, T. Zeyl, A. Senderovich and J. Beck, "Generating Complex, Realistic Cloud Workloads using Recurrent Neural

Networks.," *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles,* pp. 376-391, 2021.

[10] Y. Shao et al., "Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture," in *In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52)*, New York, 2019.

[11] G. Krishnan et al., "SIAM: Chiplet-based Scalable In-Memory Acceleration with Mesh for Deep Neural Networks," *ACM Trans. Embed. Comput. Syst,* vol. 20, no. 5, 2021.

[12] S. Bharadwaj, J. Yin, B. Beckmann and T. Krishna, "Kite: A Family of Heterogeneous Interposer Topologies Enabled via Accurate Interconnect Modeling," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020.

[13] P. Iff et al., "HexaMesh: Scaling to Hundreds of Chiplets with an Optimized Chiplet Arrangement," in *DAC*, 2023.

[14] M. Bader, "Space-filling curves: an introduction with applications in scientific computing (Vol. 9).," Springer Science & Business Media., 2012.

[15] H. Sagan, "Space-filling curves.," Springer Science & Business Media, 2012.

[16] H. Sharma et. al, "SWAP: A Server-Scale Communication-Aware Chiplet-Based Manycore PIM Accelerator," *EEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 41, no. 11, pp. 4145-4156, 2022.

[17] 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 Journal of Solid-State Circuits,* vol. 56, no. 1, 2021.

[18] T. Ebadollah, S. Pasricha and M. Nikdast, "DeFT: A deadlock-free and fault-tolerant routing algorithm for 2.5 D chiplet networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2022.

[19] E. Pete, T. Austin and V. Bertacco, "SiPterposer: A fault-tolerant substrate for flexible system-in-package design," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[20] H. Sharma et al., "Florets for Chiplets: Data Flow-aware High-Performance and Energy-efficient Network-on-Interposer for CNN Inference Tasks," *ACM Transactions on Embedded Computing Systems,* vol. 22, no. 5, pp. 1-21, 2023.

[21] D. Hilbert, "Uber die stegie Abbildung einer Linie auf Flachenstuck,," *Math. Ann.,* vol. 38 , pp. 459-460, 1891.

[22] G. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," in *IBM*, Ottawa, Canada , 1966.

[23] P. Xu and S. Tirthapura, "A lower bound on proximity preservation by space filling curves.," *IEEE 26th International Parallel and Distributed Processing Symposium,* pp. 1295-1305, 2012.

[24] S. Sarkar, G. R. Kulkarni, P. P. Pande and A. Kalyanaraman, "Network-on-chip hardware accelerators for biological sequence alignment," *IEEE Transactions on Computers,* vol. 59, no. 1, pp. 29-41, 2009.

[25] E. W. Bethel, D. Camp, D. Donofrio and M. Howison, "Improving performance of structured-memory, data-intensive applications on multi-core platforms via a space-filling curve memory layout.," in *IEEE International Parallel and Distributed Processing Symposium workshop*, 2015.

[26] M. M. Haque, A. Kalyanaraman, A. Dhingra, N. Abu-Lail and K. Graybeal, "DNAjig: a new approach for building DNA nanostructures.," in *International Conference on Bioinformatics and Biomedicine*, 2009.

[27] S. Sutardja., "1.2 the future of IC design innovation," in *IEEE Int. Solid- State Circuits Conf.*, Feb 2015.

[28] J. A. Cunningham., "The use and evaluation of yield models in integrated circuit manufacturing.," *IEEE Trans. Semicond. Manuf.,* 1990.

[29] 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 Journal of Solid-State Circuits,* vol. 55, no. 4, 2020.

[30] S. Pati, "Computation vs. Communication Scaling for Future Transformers on Future Hardware," in *arXiv*, 2023.

[31] W. Tutte, "A theorem on planar graphs.," *Transactions of the American Mathematical Society, ,* no. 1, pp. 99-116, 1956.

[32] J. E. G. Coffman, M. Garrey and D. Johnson, "Approximation Algorithms for Bin Packing — An Updated Survey.," in *Algorithm Design for Computer System Design. International Centre for Mechanical Sciences*, Springer, 1984.

[33] T. K. Hazra and A. Hore, "A comparative study of Travelling Salesman Problem and solution using different algorithm design techniques," in *7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, Vancouver, 2016.

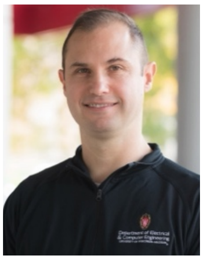
[34] X. Peng et al., "DNN+NeuroSim: An End-to-End Benchmarking Framework for Compute-in-Memory Accelerators with Versatile Device Technologies," in *International Electron Devices Meeting (IEDM)*, 2019.

[35] N. Jiang et al., "A Detailed and Flexible Cycle-Accurate Network-on-Chip Simulator," in *IEEE ISPASS*, 2013.

[36] H. Luo et al., "Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator," in *arXiv:2308.11030*, 2023.

[37] Y. Kim, W. Yang and O. Mutlu, "Ramulator: A Fast and Extensible DRAM Simulator,"," *IEEE Computer Architecture Letters,* vol. 15, no. 1, pp. 45-49, 2016.

Harsh Sharma (Student Member, IEEE) received the B.E. degree from NSIT, Delhi University, New Delhi, India, in 2021. He is currently a Ph.D. candidate at Washington State University, Pullman, USA. His research interests include His current research interests include novel interconnect architectures for manycore chips, heterogeneous architectures, and ML.



Umit Y. Ogras (Senior Member, IEEE) Umit Y. Ogras is a professor at the University of Wisconsin-Madison, Madison, WI USA. His research interests include embedded systems, heterogeneous SoCs, low-power VLSI, wearable computing, and flexible hybrid electronics. Ogras has a Ph.D. in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA.

Ananth Kalyanraman (Senior Member, IEEE) is a Professor, Boeing Centennial Chair, and the Interim Director at the School of Electrical Engineering and Computer Science, Washington State University, Pullman, Washington. He is also the Director of the AgAID AI Institute and holds a joint appointment with Pacific Northwest National Laboratory. He received PhD from Iowa State University in 2006. His research focuses on developing parallel algorithms and software for data-intensive problems in the areas of computational biology and graph-theoretic applications. Ananth serves as an Associate Editor-In-Chief of Journal of Parallel and Distributed Computing, an Associate Editor of IEEE/ACM Transactions on Computational Biology and Bioinformatics, and Subject Area Editor for Parallel Computing.

Partha Pratim Pande (Fellow, IEEE) is a professor and holder of the Boeing Centennial Chair in computer engineering at the school of Electrical Engineering and Computer Science, Washington State University, Pullman, USA. He is currently the Interim dean of the Voiland College of Engineering and Architecture (VCEA). His current research interests are novel interconnect architectures for manycore chips, on-chip wireless networks, heterogeneous architectures, and ML for EDA.