Chiplet-Gym: Optimizing Chiplet-Based AI Accelerator Design With Reinforcement Learning

Kaniz Mishty and Mehdi Sadi, *Member, IEEE*

***Abstract*—Modern Artificial Intelligence (AI) workloads demand computing systems with large silicon area to sustain throughput and competitive performance. However, prohibitive manufacturing costs and yield limitations at advanced tech nodes and die-size reaching the reticle limit restrain us from achieving this. With the recent innovations in advanced packaging technologies, chiplet-based architectures have gained significant attention in the AI hardware domain. However, the vast design space of chiplet-based AI accelerator design and the absence of system and package-level co-design methodology make it difficult for the designer to find the optimum design point regarding Power, Performance, Area, and manufacturing Cost (PPAC). This paper presents Chiplet-Gym, a Reinforcement Learning (RL)-based optimization framework to explore the vast design space of chiplet-based AI accelerators, encompassing the resource allocation, placement, and packaging architecture. We analytically model the PPAC of the chiplet-based AI accelerator and integrate it into an OpenAI gym environment to evaluate the design points. We also explore non-RL-based optimization approaches and combine these two approaches to ensure the robustness of the optimizer. The optimizer-suggested design point achieves $1.52\times$ throughput, $0.27\times$ energy, and $0.89\times$ cost of its monolithic counterpart at iso-area.**

***Index Terms*—AI accelerator, chiplet, heterogeneous integration, design space exploration, reinforcement learning.**

I. Introduction

As Large Language Models (LLMs), such as chatGPT, GPT-4, LLaMA [1], etc., gain widespread use, there is a growing demand for energy-efficient hardware that can deliver high throughput. To support hundreds of trillions of operations and hundreds of gigabytes of data movement, the high-performance and energy-efficient hardware demands more silicon area, accommodating more compute cores and memory capacity. Training any state-of-the-art AI or Deep Learning (DL) model with a single GPU or accelerator is nearly impossible due to extreme computing and memory demands. The data centers are equipped with clusters of powerful computers and GPUs connected via PCIe, NVLink, etc. [2], [3]. Even though these supercomputers can deal with large workloads, they consume a significant amount of energy [2] and involve longer latency. Because off-board communications consume at least one order of magnitude more power and time than any on-package communications [4]. The ideal scenario would be a hardware capable of housing the entire model parameters and intermediate activations on-chip [5], promising optimal performance and energy efficiency. Unfortunately, this is not feasible due to the stagnation of Moore's law and Dennard scaling, die size reaching the reticle limit, and the prohibitive manufacturing cost and yield limitations [3]. Consequently, researchers endeavor to replicate this 'hypothetical ideal' hardware concept by integrating multiple smaller chiplets at the package level, allowing near-ideal performance while minimizing costs and energy consumption.

With the advent of advanced packaging technologies, the chiplet-based heterogeneous integration has opened up a new dimension of chip design, **More-than-Moore** [3]. In chiplet-based system, multiple chiplets (i.e., SoCs) of diverse functionalities (e.g., logic dies, memories, analog IPs, accelerators etc.) and tech nodes (e.g., 7nm or beyond) from different foundries are interconnected in package level using the advanced packaging technologies, such as CoWoS, EMIB, etc. [3]. The value proposition of chiplet-based architectures is manifold. Compared to multiple monolithic SoCs interconnected via off-package or off-board links such as PCIe, NVLink, CXL etc. [3], package-level integration of multiple monolithic SoCs via 2.5D or 3D has accelerated performance and lower energy consumption alleviating off-package communications. Chiplet-based systems offer lower RE (Recurrent Engineering) cost by providing higher yield and lower NRE (Non-Recurrent Engineering) by enabling IP reuse and shortening IC design cycle [6].

The commercial chiplet-based general purpose products [7], [8] are designed and developed at vertically integrated companies without exposing much knowledge about the chiplet-based architectures' design space. Unlike these general purpose products, chiplet-based AI accelerators demand extensive design space exploration to hit the target Power, Performance, Area, and Cost (PPAC) budget. From architectural perspective, designers must consider the resource allocation, mapping and dataflow of the DNN workloads. From communication and integration perspective, chiplet placement, routing protocols, stacking/packaging technologies, interconnect types, and finally from application perspective, system requirement, such as reliability, scalibility etc., should be considered all at the same time while optimizing for PPAC [9]. The existing works often

Received 3 February 2024; revised 11 August 2024; accepted 4 September 2024. Date of publication 9 September 2024; date of current version 12 December 2024. This work was supported by the National Science Foundation (NSF) under Grant CRII-2153394. Recommended for acceptance by L. P. Carloni. *(Corresponding author: Mehdi Sadi.)*

The authors are with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849 USA (e-mail: kzm0114@auburn.edu; mehdi.sadi@auburn.edu).

Digital Object Identifier 10.1109/TC.2024.3457740

focus either on the architectural or integration aspects as a separate design flow: explore routing and packaging given chiplets [10], [11], [12] or explore chiplets architecture given the packaging [5], [13], [14], [15]. An isolated approach, addressing individual aspects independently, may result in sub-optimal designs due to the inter-dependency among these factors. For instance, varying resource allocation impacts communication demands, influencing the choice of packaging and its configuration, consequently leading to cost variations.

Currently, many flavors of packaging technologies, both from 2.5D and 3D, are available from the industry leaders, which makes it difficult for system designers and integrators to choose the optimum set of configurations from the vast design space based on the system requirements [3]. The various packaging technologies differ in fabrication cost and complexity, performance, and underlying integration technologies [3]. As a result, no single package technology can be marked as superior to others. Each of the other domains, such as resource allocation, chiplet granularity, placement, Network on Package (NoP), and interconnect architectures, to name a few, also has an extensive design space. A proper co-optimization across all these domains based on the system and application requirements at the available cost is necessary for a successful chiplet based system design. Optimizing all possible domains results in a combinatorial explosion where brute force search is not an option and random search might not result in the optimum point. The expensive simulation environment of chip design exacerbates this problem.

To overcome these limitations, in this paper, we make the following contributions bridging the gap between the system requirements and design aggregation, planning, and optimization for chiplet-based architecture.

- We develop a co-design methodology for chiplet-based AI accelerators. The co-design task contemplates resource allocation, such as the number of AI chiplets, memory capacity, and bandwidth; partitioning and placement of chiplets such as aspect ratio of the accelerator chiplet arrays, and logical placement of accelerator and memory chiplet; different packaging technologies (i.e., CoWoS, EMIB, SoIC, and FOVEROS [3]) and their attributes such as bandwidth, bump pitch density, cost and complexity, to optimize the system-level Power, Performance, Area, and Cost (PPAC) of the chiplet-based AI accelerators.
- We formulate an analytical cost model for assessing the chiplet-based architectures. This analytical model enables us to assess the chiplet-based AI accelerator in a time-and-resource-constrained environment.
- To optimize throughput, energy efficiency, and cost, we identify the inter-dependency of the design space parameters and formulate the optimization problem as a Reinforcement Learning (RL) problem. We also explore non-RL based optimization approaches, such as simulated annealing, and combine these two approaches to ensure the robustness of the optimizer.
- Finally, we validate our methodology by comparing the performance of our optimized design against state-of-the-art monolithic GPU on MLPerf benchmark and justify the performance improvement.

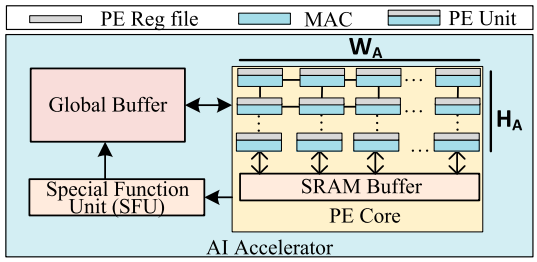

Fig. 1. AI accelerator chiplet architecture.

The rest of the article is organized as follows. Section II presents the background. Section III describes the analytical modeling and design space exploration. The optimization framework is presented in Section IV followed by experiments and results in Section V, related works in Section VI, limitations and future works in Section VII and conclusion in Section VIII.

II. Background

A. AI Workloads and Accelerators

1) AI Workloads: The primary domains of AI encompass Computer Vision (CV), Natural Language Processing (NLP), Recommender Systems, and Reinforcement Learning. The integration of these domains has led to the emergence of Generative AI, enabling models to generate diverse content, including text and images. In Generative or Multi-modal AI, diverse AI/DNN (Deep Neural Network) models are fused together to generate an output. While the architectural characteristics and parameters of LLM and CV models may differ, their fundamental components share similarities with the structure of Transformer [16] for NLP and ResNet [17] for CV, respectively. The critical operations in CV models involve regular convolution, Depth-wise or Point-wise convolution, residual blocks, FC (Fully Connected) operations, whereas the scaled-dot product attention operations, and FC operations dominate in LLM. These operations can be expressed as or converted to matrix-matrix/vector multiplication (GEMM) with massive parallelism.

2) AI Accelerator: Systolic array [18] type architecture, leveraging the inherent parallelism of DNN workloads, has been used as the core of AI accelerators. A typical AI accelerator is composed of arrays of Processing Element (PE) for computation and on-chip buffer to hold the weights and activations. PEs are composed of Multiplier-Adder (MAC) units and small register file for each MAC units to hold the stationary data, depending on the dataflow. The size of PE array, memory hierarchy, and memory size are critical design parameter of a AI accelerator. Fig. 1 shows a AI accelerator with a PE core, Special Function Unit (SFU), and Global Buffer. The PE core contains a small SRAM buffer and bunch of PE units. Each PE consists of a MAC unit and a reg. file [19].

B. Chiplets and Heterogeneous Integration

1) 2.5D Architecture: In 2.5D architecture, two or more chiplets, fabricated separately, are connected side-by-side with each other in package-level through interposer (silicon/organic) or silicon bridge. Two commercial 2.5D interconnects are Chip on Wafer on Substrate (CoWoS) from TSMC [20] and Embedded Multi-die Interconnect Bridge (EMIB) from Intel [21]. In

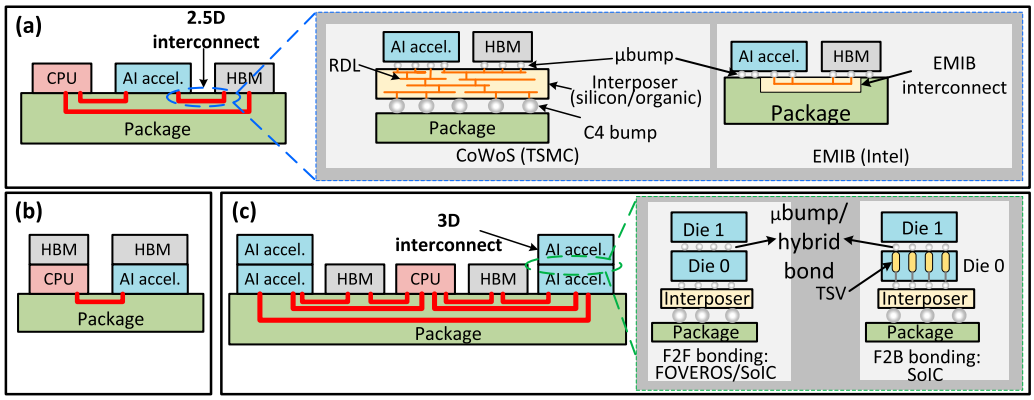

Fig. 2. Top-level system architecture for different scenarios. (a) CPU, AI accelerator and HBM chiplets are connected in package level through 2.5D interconnects. CoWoS and EMIB are two options of 2.5D interconnects. (b) CPU and AI accelerator chiplets are connected through 2.5D interconnects and HBM is stacked on top of CPU and AI accelerator through 3D interconnects. (c) Two AI accelerator chiplets are stacked on top of each other through 3D interconnects and they are interconnected to CPU, HBM and other AI chiplets pair through 2.5D.

CoWoS, two side-by-side dies are connected with each other and with package substrate through an intermediate interposer layer [20].

Interposer can be active and passive. Active interposer contains embedded logics and Re-Distribution Layers (RDL) where as passive interposer containing RDLs are only used for routing purpose. CoWoS typically employs passive interposer for 2.5D integration. In contrast, Intel's EMIB utilizes thin silicon pieces with multilayer BEOL interconnects (Silicon Bridge) embedded in the organic package substrate for high-density localized interconnects, eliminating the need for a separate interposer layer [21]. CoWoS and EMIB architectures are illustrated in Fig. 2(a).

2) 3D Architecture: In 3D, two or more separately fabricated chiplets are stacked on top of each other through 3D interconnects formed with copper micro-bumps, or hybrid wafer bonding [22]. Depending on the bonding interface orientation of the interconnected dies, different bonding configurations are possible, such as face-to-face (F2F), face-to-back (F2B), back-to-back (B2B) etc. Intel's FOVEROS [23] uses F2F bonding where the face of the top die is bonded to the face of the bottom die (active interposer) through Cu micro-bump connections. Bottom die is connected to the package through TSV [23]. TSMC has the option of both F2F and F2B bonding configuration in their System on Integrated Chips (SoIC), however, they use hybrid bonding instead of Cu μ-bumps [24]. The latest upgrade of FOVEROS, FOVEROS-Direct, also leverages direct cu-cu hybrid bonding for inter-die interconnection. Recently, both 2.5D and 3D can be integrated on the same package and these architectures are known as 5.5D [25].

III. Throughput Formulation and Design Space Exploration

In this section, we formulate the cost model for chiplet-based AI accelerators, including throughput, energy, and cost. We perform design space exploration to comprehend the influence of various design parameters on the cost model.

A. Top Level Architectural Exploration

We explore two architectural approaches: (i) 2.5D architecture, where all chiplets are connected with each other at the package level through 2.5D interconnects (Fig. 2(a)). (ii) 5.5D (combining 2.5D and 3D) [25] where two or more 3D-stacked (connected via 3D interconnects) chiplets are further linked through 2.5D interconnects (Fig. 2(b) & (c)). In all cases, the architecture of the AI accelerator chiplet is a regular systolic-array composed of PE array and dedicated on-chip buffer shown in Fig. 1 [19]. However, the number of PE units and on-chip buffer size varies with the number of allocated chiplets, as we consider a fixed package size.

1) 2.5D Architecture: In 2.5D architecture, we consider that CPU, AI accelerator, and HBM chiplets are connected at the package level through 2.5D interconnects (Fig. 2(a)). We explore two 2.5D integration technologies, EMIB and CoWoS, and their different configurations.

2) 5.5D Architecture (Combining 2.5D and 3D): 5.5D architecture is divided into two cases: (i) memory-on-logic, where HBMs are stacked on top of CPU and/or AI chiplets as shown in Fig. 2(b), and (ii) logic-on-logic, where two AI chiplets are 3D-stacked on top of each other. These 3D-stacked AI chiplets are connected to CPU and/or HBM and other 3D-stacked AI chiplets through 2.5D interconnects as shown in Fig. 2(c). To avoid temperature-induced breakdowns [22], we limit our exploration to only 2-tiers. We explore the off-the-shelf 3D integration techniques, SoIC and FOVEROS, and their different configurations. Depending on the integration technology and their configuration settings, these architectures offer different bandwidths, energy efficiency, area efficiency, and cost.

B. Throughput and Energy Efficiency Formulation

1) Throughput: We define system throughput as *tasks completed per second*,

$$T = \frac{tasks}{sec} \tag{1}$$

$tasks$ represents different entities depending on the DNN domain and its mode of operations. During inference, $tasks$ represents the number of $inferences$. During training of CV and NLP models, $tasks$ means the number of $images$ and $tokens$ processed per second, respectively. $tasks/sec$ can be decomposed into [26]

$$\frac{tasks}{sec} = \frac{ops}{sec} \times \frac{1}{(\frac{ops}{task})_G} \times \frac{1}{(\frac{ops}{task})_{nG}} \times M_{eff} \quad (2)$$

Here, ops/sec depends on both DNN hardware and DNN models. GEMM operations per task, $(ops/task)_G$, and non-GEMM operations per task $(ops/task)_{nG}$ depend on only DNN models, and M_{eff}, mapping efficiency depends on DNN models and hardware, along with mapping strategies. ops means the MAC operation. The GEMM operations are performed in the systolic array. The non-GEMM operations such as softmax is performed in the SFU of the accelerator. Dropout and residual operations, manifested as Element-wise multiplication and addition, is also performed using the MAC modules. Layer normalization and other reduction or control flow operations are taken care in the ALU or scalar unit of the SFU.

For a system comprising multiple AI accelerator chiplets, the $operations/sec$ is expressed as,

$$\left(\frac{ops}{sec}\right)_{sys} = \left(\frac{ops}{sec}\right)_{AI\_chip} \times AI\_chip_{tot} \times U_{sys} \quad (3)$$

Where $(ops/sec)_{AI\_chip}$ is the peak throughput per AI chiplet, AI_chip_{tot} = total number of AI chiplets, and U_{sys} = system utilization factor. It represents the effective fraction of the active chiplets out of the total chiplets. It depends on the interchiplet communication bandwidth (BW_{AI-AI}), determined by choice of the packaging architecture, package type, and their different configuration. In section III-D1, we describe this in detail. The peak throughput per AI chiplet is expressed as

$$\left(\frac{ops}{sec}\right)_{AI\_chip} = \left(\frac{1}{\frac{cycles}{op}} \times \frac{cycles}{sec}\right) \times PE_{tot} \times U_{AI\_chip} \quad (4)$$

Where,

$$\frac{cycles}{op} = cycle_{comm} + cycle_{op^*} \quad (5)$$

$cycle_{comm}$ = chiplet-to-chiplet communication latency, $cycle_{op^*}$ = arithmetic operation latency of the chiplet microachitecture, and $cycles/sec$=f, frequency of the AI accelerator chiplets. $cycle_{comm}$ depends on the distance between the data source and destination. It is impacted by the chiplet allocation, chiplet array dimension (i.e., number of AI chiplets in X and Y dimension) and the physical location of the AI and HBM chiplets. $cycle_{op^*}$ depends on the microarchitecture of the chiplet (design of PE array, MAC unit) and the type of operations. We assume that all AI chiplets can operate at the same frequency and have the same architectural and functional configuration. However, the frequency of each chiplet can be further controlled based on the data traffic and location of chiplets to optimize system throughput and energy. PE_{tot} = total number of PEs per AI chiplet, and U_{AI_chip} = chiplet utilization representing the fraction of PEs utilized during computation. U_{AI_chip} depends on mapping of the AI model tasks to the accelerator.

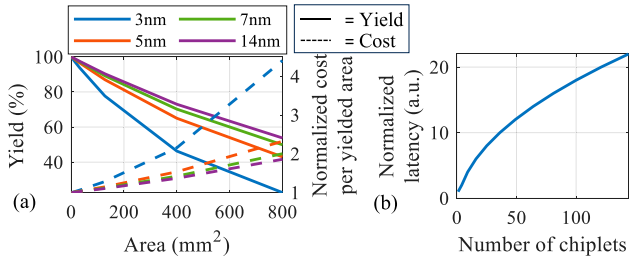

Fig. 3. (a) Yield (left y-axis) and normalized cost per yielded area (right y-axis) vs area at different tech nodes. (b) Normalized latency vs number of chiplets.

2) Energy Efficiency: Energy efficiency is paramount when processing DNN at edge devices and cloud data centers. Edge devices are usually constrained by battery life and thermal budget, and data centers are typically constrained by electricity bills, thermal budget, and environmental impact [2]. Data centers are mainly focused on achieving higher throughput, which requires higher energy budget. In this work, we closely monitor energy efficiency while maximizing the throughput.

We define energy efficiency (E_{eff}) of a system as *tasks completed per joule*:

$$E_{eff} = \frac{tasks}{joule} = \frac{1}{\frac{joules}{ops}} * \frac{1}{\frac{ops}{task}} \quad (6)$$

$joules/operations$ depends on both DNN hardware and DNN models, whereas $operations/task$ depends only on the considered DNN model. We break down the energy per operations, E_{op}, (i.e. $joules/operations$) into its constituent parts:

$$E_{op} = E_{comm} + E_{op^*} \quad (7)$$

E_{comm} is the energy required to transfer data from chiplet-to-chiplet and E_{op^*} is the energy to perform an arithmetic operation. E_{comm} depends on the choice of packaging architectures (e.g., 2.5D, 3D) and interconnect types (e.g., EMIB, CoWoS, FOVEROS, SoIC) and E_{op^*} depends on the microarchitecture.

C. Chiplet Allocation and Placement

In the context of chiplet-based accelerator design, determining the number of chiplets, area allocated to each chiplet, and their placement becomes pivotal, as they impact the throughput, energy, and cost. Here we will delve into the relationship between yield, area, cost, communication latency across various chiplet configurations.

1) Yield and Cost vs Area: Intuitively, as the chip area increases, its compute and memory capacity increases, ensuring high performance and energy-efficiency. However, as shown in Fig. 3(a), we are limited by the fact that in advanced tech nodes, as the chip area increases, yield decreases, resulting in increased cost per area [6]. The yield of the manufactured chip, Y_{die} is expressed as the following Negative Binomial model:

$$Y_{chip} = \left(1 + \frac{dA}{\alpha}\right)^{-\alpha} \quad (8)$$

where d is the defect density of the tech node, A is the area of the chip, and α is the cluster parameter. Assuming P_0 as unit price, we can also estimate the cost per yielded area as

$$C_{yield} = \frac{P_0}{Y_{chip}} \approx P_0(1 + dA + \frac{\alpha - 1}{2\alpha} d^2 A^2) \quad (9)$$

2) Inter-Chiplet Communication Latency: As mentioned earlier, the chiplet-to-chiplet data communication latency, $cycle_{comm}$, impacts the system performance by contributing to $cycles/operations$. Data transfer between chiplets occurs through the package-level interconnects such as CoWoS, EMIB, FOVEROS, SoIC etc. Considering that the data might be supplied from another AI-chiplet or directly from HBM, we estimate both AI-AI chiplet communication latency, L_{AI-AI}, and HBM-AI communication latency, L_{HBM-AI}.

$$cycle_{comm} = \begin{cases} L_{AI-AI} & \text{if data moves from AI to AI chip} \\ L_{HBM-AI} & \text{if data moves from mem. to AI chip} \end{cases} \quad (10)$$

Impact of AI chiplet count. As the number of chiplet increases, the physical distance between the source and destination chiplet increases, resulting in increased communication latency. We consider 2D-mesh topology, which is widely used in tile-based architecture for its simplicity and scalability. Routing in the package substrate is more intricate than on-chip. As a result, tile-based chiplet architectures have been architected with mesh topology [13]. Fig. 3(b) shows that communication latency increases drastically with the number of chiplets for a mesh topology.

Impact of Chiplet array dimension. The longest AI-to-AI chiplet communication latency is expressed as

$$L_{AI-AI} = H_{AI-AI} \times t_w + H_{AI-AI} \times t_r + T_c + T_s \quad (11)$$

As we consider a 2D mesh of AI accelerator chiplets, $H_{AI-AI} = m + n - 2$ denotes the number of hops between the source-destination pair. m, n represent the number of AI chiplets in the X and Y dimension of the array, respectively. t_w is per-hop wire delay, t_r, T_c, and T_s are router delay, contention delay, and serialization delay, respectively [27]. Here, t_w, t_r, T_s are design time metrics, that depend on tech. node, interconnect technologies, circuit, and microarchitecture design, T_c depends on workload/data traffic. For a fixed number of chiplets and routing topology, H_{AI-AI} depends on the chiplet array X and Y dimension. We try to keep the aspect ratio of the chiplet array as close as possible to 1 to reduce the communication latency. In addition, the physical dimension of the chiplet array impacts the system performance by affecting the choice of dataflow and workload mapping strategies [14]. For a fixed dataflow and mapping strategy, the system performance largely depends on the chiplet array dimension as shown in Fig. 4.

Impact of HBM/CPU count and location. We analyze the impact of dividing the allocated HBM into multiple chiplets and placing the chiplets in multiple positions on system latency. Partitioning a large chunk of memory into multiple memory chiplets (instead of placing the large memory in one place) and placing these multiple memory chiplets in different locations improves the system latency. Unlike, AI chiplet counts, as the number of HBM chiplets increases, communication latency decreases. Because the communication latency depends on the physical location of the data [13]. Fig. 4 illustrates how chiplet partitioning and placement improve the system latency. As we consider a 2D mesh of AI accelerator chiplets, there are 6 locations: left, right, top, bottom, middle, and 3D stacking, to place the HBM chiplets around the AI chiplets array. These locations result in $2^6 - 1$ combinations for HBM/CPU placements. We model $L_{HBM/CPU-AI}$ same as equation 11, where H_{AI-AI} is replaced by $H_{HBM/CPU-AI}$. We use the model presented in [28] to calculate $H_{HBM/CPU-AI}$ for different locations of HBM/CPU pair. We consider a 16GB (8-stack, each stack 16Gb) HBM3 chiplet [29], giving the highest capacity of 80GB with 5 chiplets. We assume that each HBM chiplet has a dedicated memory controller and NoC router integrated within it [30]. As a result, at iso-memory-capacity (i.e, same number of HBMs with integrated memory controller) the cost associated with HBM for both monolithic and chiplet systems is equivalent.

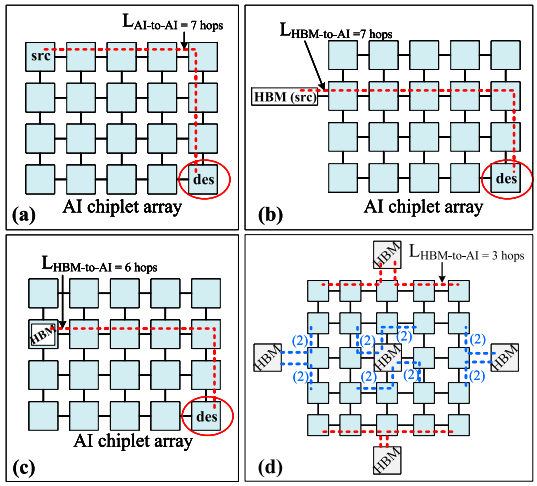

Fig. 4. Illustration of latency (in terms of hop) calculation. (a) AI2AI chiplet communication, considering the farthest chiplets as source-destination pair. (b) One HBM chiplet, located at the left connected in 2.5D, and the farthest AI chiplet as source-destination pair. (c) One HBM chiplet, 3D-stacked on top of a left-most AI chiplet, and the farthest AI chiplet as source-destination pair. (d) 5 HBM chiplets are placed in 5 different positions. The highest latency decreases from 6 hops (case (c)) to 3 hops with most of the AI chiplets can be provided with data in 2 hops by nearest HBMs.

The host CPU is primarily responsible for dispatching the workloads to the accelerator chiplets. The package area is shared by accelerator chiplet, HBMs as well as CPUs. However, the majority of the package area is used for AI computing and HBM memories [13], [30]. Hence, in this work we only focus on the AI accelerator and HBMs.

The above discussion suggests that, for cost-effective integration of more functionalities, we should partition the total chip area into multiple chiplets, each with smaller areas. From the yield and cost perspective, the more the number of chiplets, the better throughput and less cost. However, this also introduces

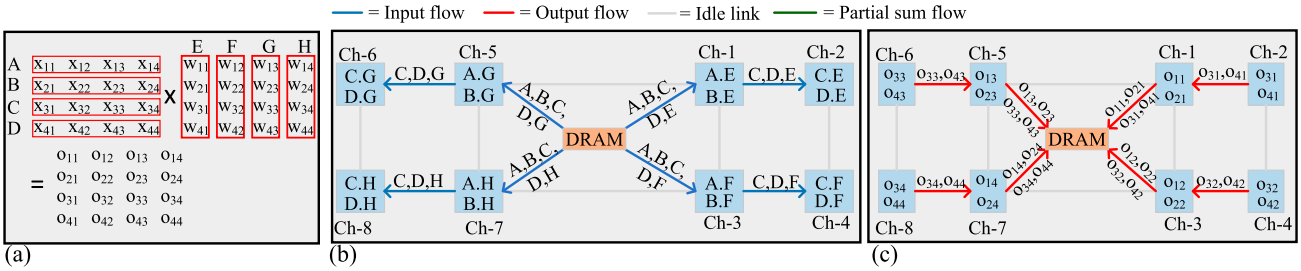

Fig. 5. Illustration of mapping and dataflow. (a) Splitting the matrices into smaller parts for different chiplets. (b) Initial data supply from DRAM. Once the chiplets are loaded with required data, computation begins. (c) Final output collection to the DRAM. In this dataflow, there is no inter-chiplet communication during computation for partial sum.

another consideration: an increase in the number of chiplets results in higher inter-chiplet communication latency, ultimately diminishing throughput and energy efficiency. Therefore, a balance must be struck between dividing the area into an appropriate number of chiplets to enhance functionality and ensure the associated communication latency does not compromise overall system performance and efficiency.

D. Package Architectures and Configurations

We explore different packaging architectures, interconnects, and their different configurations [20], [21], [23], [24] to analyze their impact on the system performance and budget.

1) Inter-Chiplet Communication Bandwidth: The system utilization term, U_{sys}, of equation 3 depends on the inter-chiplet communication bandwidth. We define U_{sys}:

$$U_{sys} = \frac{BW_{act}}{BW_{req}} \tag{12}$$

Where, BW_{act} is the actual bytes of data transferred per sec and BW_{req} is the required bytes to keep all the neighboring AI chiplets at 100% utilization, i.e., no stalling for data. For the layout of AI and HBM chiplets we consider in this work, the HBM chiplet needs to deliver data to its 4 neighboring AI chiplets simultaneously at most, and any AI chiplet needs to deliver data to its 1 neighboring chiplets at most. However, it can change with the mapping strategies. As the communication between CPU and AI chiplet primarily involves the instruction dispatch and output accumulation, the communication bandwidth is dominated by bandwidth requirements of the AI accelerator to HBM chiplet.

Chiplet mapping exploration. For large sequence lengths and batch sizes of NLP/LLM models as well for large FC/Conv. layers of DNN models, the matrix sizes get larger, which need to be split temporally in the monolithic chips if the monolithic chip does not contain enough PE units and memory. Having multiple chiplets, the matrices can be split spatially and mapped to multiple chiplets, performing parallel computation. As illustrated in Fig. 5(a-c), the input matrix is split along rows (A, B, C, D), and the weight matrix is split along columns (E, F, G, H). Chiplets 1, 3, 5, 7 handle data chunks A and B, while Chiplets 2, 4, 6, 8 handle C and D. The weight matrix portions (E, F, G, H) are distributed to all chiplets accordingly. During initialization, the DRAM supplies data $4 \times [A, B, C, D]$, and $[E, F, G, H]$ simultaneously to chiplets 1, 3, 5, 4, with A and B reaching neighboring chiplets in one hop and C and D reaching distant chiplets in the next hop. Data chunks E, F, G, H reach neighboring and distant chiplets in one hop and two hops, respectively. The outputs are collected back to DRAM once the computations are completed. Outputs from neighboring chiplets (ch-1, ch-3, ch-5, ch-7) reach DRAM in one hop, while outputs from distant chiplets reach DRAM in two hops. No inter-chiplet communication is required for partial sum accumulation, however, the required AI-HBM bandwidth (or the number of channels) is higher in this mapping strategy, as DRAM needs to broadcast $[A, B, C, D]$ to all four neighboring chiplets. According to the above mentioned mapping and dataflow, the required bandwidth is formulated as

$$BW_{req} = \begin{cases} 4 \times N_o \times d_w \times f \times (\frac{ops}{sec})_{AI\_chip} & \text{if src. is HBM} \\ 1 \times N_o \times d_w \times f \times (\frac{ops}{sec})_{AI\_chip} & \text{if src. is AI chip} \end{cases} \tag{13}$$

Where, N_o is the number of operands required to perform a MAC operation, which is 2 in general (two multipliers for the multiplication and no new external operands are needed for addition). d_w is the data width and $(ops/sec)_{AI\_chip}$ is the peak throughput of the AI chiplet, and f is the frequency of the accelerator. If $BW_{act} \geq BW_{req}$, then there is no stalling in initializing the chiplets' PE array with data. However, if $BW_{act} < BW_{req}$, then there will be $\lceil \frac{BW_{req}}{BW_{act}} \rceil$ cycle stalling for operand data to start the computation. We penalize the overall system throughput with these stalling periods while estimating the system throughput. From equation 13, the required bandwidth is smaller if the peak throughput of the AI chiplet is low, resulting in less penalty.

Impact of Data rates and Link count. The data rate per pin (in Gbps), DR, and the number of links assigned for data transfer, L, of different package type determine the active bandwidth, BW_{act},

$$BW_{act} = DR \times L \tag{14}$$

DR and L depend on the interconnect technology. It plays a significant role in the system throughput by contributing to the system utilization.

TABLE I
PARAMETERS AND VALUES OF DESIGN SPACE

Parameter	Values
Architecture type	2.5D, 5.5D: (i) memory-on-logic (ii) logic-on-logic
No. of chiplets	1 to 128 @ step of 1
No. & location of HBMs	Left, right, top, bottom, middle, 3D stacked; 2^6 -1 location
AI2AI interconnect 2.5D	CoWoS, EMIB
AI2AI data rate 2.5D (Gbps)	1 to 20 @ step of 1
AI2AI link count 2.5D	50 to 5000 @ step of 50
AI2AI trace length (mm) 2.5D	1 to 10 @ step of 1
AI2AI interconnect 3D	SoIC, FOVEROS
AI2AI data rate 3D (Gbps)	20 to 50 @ step of 1
AI2AI link count 3D	100 to 10,000 @ step of 100
AI2HBM interconnect 2.5D	CoWoS, EMIB
AI2HBM data rate 2.5D (Gbps)	1 to 20 @ step of 1
AI2HBM link count 2.5D	50 to 5000 @ step of 50
AI2HBM trace length 2.5D (mm)	1 to 10 @ step of 1

2) Inter-Chiplet Communication Energy: Interchiplet communication energy E_{comm} depends on the packaging architecture and the data transfer volume. We model it as

$$E_{comm} = E_{bit\_pkg} \times bit_{tot} \tag{15}$$

E_{bit_pkg} is the energy per bit data communication for different interconnect technologies, and bit_{tot} is the data traffic required for the desired operation.

Impact of trace length and no. of RDL layers. For a specific data rate and link count, E_{bit_pkg} again depends on trace length, tr_len, (link-to-link distance between two interconnected dies). To achieve a specified data rate over a longer trace length, intricate circuit techniques and more RDL layers are required resulting in the $E_{bit\_pkg} \propto tr\_len$ relationship [20].

3) Packaging Cost: The packaging cost (C_P) depends on the packaging architecture and interconnect type. For the same package type, the packaging cost again depends on (i) package area (A_P), (ii) number of layers (i.e., core and RDL), and (iii) link count (L) and modeled as [31]:

$$C_P = \mu_0 A_P + \mu_1 L + \mu_2 \tag{16}$$

Where μ_0, μ_1, and μ_2 are the regression parameters based on the number of core and RD layers. In this work, we consider a fixed package area of 900mm^2, leaving the packaging cost dependent on the number of package layers and link density.

The above discussion suggests that, based on the BW_{req}, which also depends on the number of chiplets, energy and cost budget, appropriate allocation of DR and L requires co-optimization, such that the hardware is not suffering from under-utilization while not spending too much budget unnecessarily.

IV. OPTIMIZING CHIPLET-BASED ARCHITECTURE

In this section, we build a framework to efficiently navigate the search space, as detailed in Table I, aiming to optimize throughput, energy, and cost efficiency.

Comprising of 14 parameters and their possible values, our parameter space has more than 2×10^{17} design points which poses challenges for exhaustive search due to its time and resource-intensive nature. To address this, we explore learning-based and meta-heuristic search approaches to efficiently reach global or near-global optima.

Because of the inherent stochastic nature of Reinforcement Learning (RL) and Simulated Annealing (SA) algorithms, we observe slight variations in the achieved objective function values. To enhance the robustness of the optimizer, we train multiple RL models and SA algorithms with different seed values. Subsequently, we perform an exhaustive search across the outcomes of these algorithms to pinpoint the optimum solution (refer to Alg. 1). An overview of the optimization framework is presented in Fig. 6. It takes the design space and constraints as input and outputs the optimized design points.

Algorithm 1: Proposed optimization algorithm

```
t ← Trial_max;
obj_best ← −inf;
while t ≤ Trial_max do
    param_SA, obj_SA ← SA();
    if obj_SA > obj_best then
        param_best, obj_best ← param_SA, obj_SA;
    end
    param_RL, obj_RL ← RL();
    if obj_RL > obj_best then
        param_best, obj_best ← param_RL, obj_RL;
    end
end
return param_best, obj_best
```

A. RL Problem Formulation

RL tries to mimic human learning behavior to learn about a new environment. In RL, an **Agent** continuously interacts with an **Environment**, takes **Actions** by observing the present **State** of the environment, receives feedback as a form of **Reward** from the environment, and updates its underlying **Policy** to take new actions to maximize reward. After enough interaction with the environment, the agent can take a specific set (sequence) of actions that maximize the reward in the given environment. Formulating a Markov Decision Process (MDP) consisting of a tuple of five key elements: $< \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \delta >$ is at the core of formulating an RL problem. Where $\mathcal{S}$ = State space, $\mathcal{A}$ = Action space, $\mathcal{P}$ = Transition probability matrix of going to S_t from S_{t-1} by taking action A_{t-1}, r = Reward function, and δ = discount factor that takes any value between $[0, 1]$ [32].

Environment provides feedback to the agent by quantifying the rewards. In our case, we incorporate our analytical expressions discussed in Section III into a Gym [33] environment, known as Chiplet-Gym, to assess the performance of the action taken by the agent.

State or Observation space contains the set of all possible states of the environment. It should include all the information for the agent to take the next action, making the process an MDP. In our case, the observation space contains the following items: {*maximum package area, the maximum area allowed per chiplet, current area per chiplet, ai2ai communication latency, ai2hbm communication latency, current communication energy, current packaging cost, current throughput*}.

Action space defines the set of all possible actions available to the agent each time step. Our action space, consisting of a

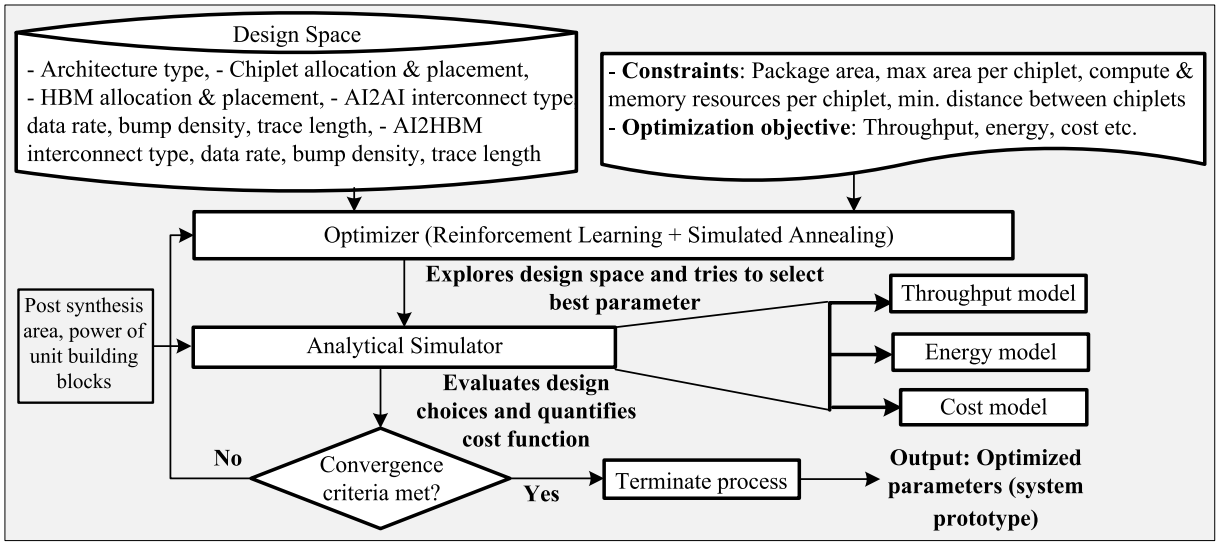

Fig. 6. Optimization framework overview.

combination of discrete integers and categorical values, corresponds to the parameters we aim to optimize. Given the state of the environment and the reward, the agent selects values for each of the parameters in Table I to maximize the reward.

Reward is provided to the agent as a form of feedback in response to every action it takes. We formulate the reward function same as the objective function we want to maximize

$$r = \alpha T - \beta E - \gamma C \tag{17}$$

Where T, E, C represent the throughput, communication energy, and packaging cost respectively. α, β, γ are the user-defined constants that let the users put specific weight on specific parameters of the objections function, such as throughput, cost, energy-efficiency during optimization. Based on the reward, which is formulated from the analytical expressions of Section III, RL finds the optimum design choices considering complex trade-offs of chiplet area, bandwidth, chiplet-to-chiplet communication.

RL algorithm We use Proximal Policy Optimization (PPO) algorithm [34] implemented by Stable-Baselines3 [35] because of its simplicity, computational efficiency, and compatibility with the action and state space of our problem. PPO is a on-policy policy gradient method that combines the idea of having multiple workers from Advantage Actor-Critic (A2C) algorithm and the idea of using trust region to improve the current policy from Trust Region Policy Optimization (TRPO) algorithm [35].

B. Simulated Annealing

In addition to RL, we also explore meta-heuristic search approaches, such as simulated annealing, to evaluate their efficacy in navigating the design space. Simulated annealing adds an exploitation step on top of random search. It randomly samples the design points and in addition to accepting the better design points, based on the acceptance criterion, it also accepts the design points that worsen the objective function. We modify the simulated annealing algorithm by slightly changing the acceptance criterion for our problem. The algorithm is shown in Algorithm 2. We optimize the same objective function as shown in Equation 17.

Algorithm 2: Modified simulated annealing algorithm

$iteration \leftarrow T_{max}$;
$temp \leftarrow temperature$;
$st\_sz \leftarrow step\_size$;
$X_{curr} \leftarrow$ randomly choose initial solution;
$O_{curr} \leftarrow$ evaluate initial solution;
$X_{best}, O_{best} \leftarrow X_{curr}, O_{best}$;
while $iterations \leq T_{max}$ **do**
  /* find candidate solution */
  $X_{cand} \leftarrow X_{curr} + uniform(-1, 1) * st\_sz$;
  /* evaluate candidate solution */
  $O_{cand} \leftarrow f(X_{cand})$;
  if $O_{cand} > O_{best}$ **then**
    $O_{best} \leftarrow O_{cand}$;
    $X_{best} \leftarrow X_{best}$;
  end
  $t \leftarrow temp/iterations$;
  if $O_{cand} > O_{curr}$ ***OR*** $rand() < t$ **then**
    $X_{curr}, O_{curr} \leftarrow X_{cand}, O_{cand}$;
  end
end
return X_{best}, O_{best}

Finally, we deploy RL and SA algorithm multiple times, followed by conducting a comprehensive search on the outputs produced by SA and RL agents.

While demonstrated explicitly for AI accelerators and mesh routing topology, the proposed optimization framework can be generalized to diverse chiplet-based designs and routing topology, requiring users to model their architectures and network topology in equation 4, 10, 11, and 13, to find the correct blend of package and interconnect architecture. For example, I/O chiplets provide signal transmission and regeneration. Their performance can be modeled as extra latency in our framework.

TABLE II
PER HOP WIRE LENGTH AND DELAY FOR 2.5D AND 3D ARCHITECTURE [21], [36]

Packaging arch.	Per hop wire length (mm)	Delay, t_w (ps)
2.5D	1	17.2
3D	0.08	1.6

TABLE III
INTERCONNECTS' PROPERTIES [20]

Interconnect	Bond/bump pitch (μm)	TSV pitch pitch(μm)	Energy (pJ/bit)	Implementation cost
CoWoS	30 - 40	-	0.2 ∼0.5	Medium
EMIB	55 - 45	-	0.17 ∼0.7	Low
SoIC	9	9	0.1 ∼0.2	High
FOVEROS	< 10	-	< 0.05	Highest

V. EXPERIMENTS AND RESULTS

A. Experimental Method

As shown in Fig. 6, at the core of the optimizer we implement PPO and simulated annealing algorithm. The optimizer explores the design space and tries to select the best parameters sticking to the design constraints and user-given optimization objective, such as throughput optimization, energy, and/or cost optimization. To evaluate the optimizer's objective function, we implement our cost model, explained in Section III, in an OpenAI Gym [33] environment named as Chiplet-Gym.

We consider a fixed amount of package area, 900mm^2, dedicated for AI and HBM chiplets [30]. To avoid thermal hotspot, we place the chiplets at 1mm apart from each other in a mesh topology [37]. This leaves $(900 - (m + n + 2)mm^2)$ of area for the chiplets. The optimizer will select the number of chiplets such that it maximizes the throughput while sticking to the area constraint. The area per chiplet is calculated as the total package area available for AI chiplets over the number of chiplets. Analyzing the yield vs area curve (Fig. 3) we set the maximum allowable area per chiplet to 400mm^2 as a constraint. Because, at 14nm, for the die area beyond 400mm^2 the yield is even lower than 75%. Inspired by the recent trend of higher on-chip memory to reduce the DRAM accesses [18], we allocate 40% of the chiplet area to the compute resources, 40% to the on-chip SRAM, and rest 20% to other blocks such as control, IO, NoC, routing etc. For 3D architecture, we have to sacrifice some of the area of the chiplet for the TSV and its associated keepout zone. From SoIC TSV pitch of 9um [20], >12K TSVs can be fit into 1mm^2. So we keep at most 2mm^2 for TSV in 3D architecture. Which is enough for both signal and power supply [38]. We use the values shown in Table II and III in our throughput, cost, and energy model to calculate the cost function of the design points.

B. Implementation Details

The optimization framework[1] is written in Python v3.9. and run on an Intel hexa-core i5-9500 @ 3 GHz machine.

1) RL: The Chiplet-Gym environment is constructed by integrating our analytical simulator into OpenAI Gym v0.26.2 [33] to establish a unified interface between the RL algorithm and the analytical simulator. We define the action space as MultiDiscrete and observation space as Box space. The simulator receives the RL policy's action (i.e., a combination of various parameters forming a design point) as input and produces corresponding throughput, energy, and cost values. The environment's state is then updated, and the reward is calculated. The state and reward are fed to the agent, enabling it to adjust its network to maximize rewards for subsequent actions.

[1]https://github.com/KFM135/chiplet-optimizer

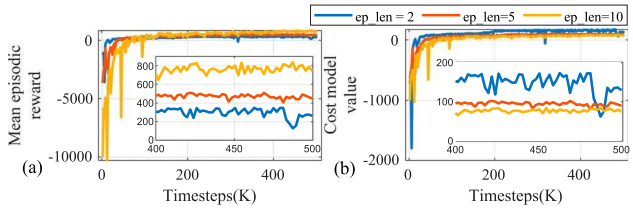

Fig. 7. Impact of episode length in convergence (PPO algorithm). Inset shows the zoomed-in version of each plot.

Policy-Value network. PPO utilizes the Multi-Layer Perceptron (MLP) as both of its policy and value network. The architecture of the actor or policy network is defined as [10, 64, 64, 810], and the architecture of the critic or value network is set as [10, 64, 64, 1], employing the tanh activation function. The size of the input for both networks is determined by the dimension of the observation space, while the output layer size of the policy network is determined by the action space. The output layer size of the value network is set to 1.

Impact of episode length on RL convergence. The algorithms are trained with an episode length of 2. While a longer episode length often results in a higher mean episodic reward, it does not guarantee a superior cost model value for the optimized parameters. Although longer episodes are generally associated with increased exploration, our hypothesis is that, in our specific case, the agents lean towards exploitation to maximize rewards. This hypothesis arises from the fact that our reward values span from a large negative value to a positive one. Once the agent discovers a positive value, it tends to exploit that particular action to maximize the mean episodic reward neglecting further exploration of the design space. Figure 7(a) shows that the agent achieves a mean episodic reward of 800 at episode length of 10, where as the cost model value of these actions are less than 100 (Fig. 7(b)). On contrary, at episode length of 2, the mean episodic reward is around 300 and the cost model value is around 150. (Note: The cost model value at each timesteps are calculated as $mean\_episodic\_reward/episode\_length$.)

Impact of entropy coefficient on RL convergence. Another hyper-parameter impacting the exploration and exploitation balance is entropy coefficient. Serving as a regularizer, entropy coefficient plays a crucial role in shaping the behavior of the RL agent during training. A larger entropy coefficient implies that all actions are equally likely, fostering exploration, while a smaller entropy coefficient indicates that one action's probability within the policy dominates, emphasizing exploitation. Fig. 8(a) shows that when entropy coefficient is set to 0, the agent stabilizes to a lower reward value more rapidly. However, when the entropy coefficient is increased to 0.1, the agent achieves a

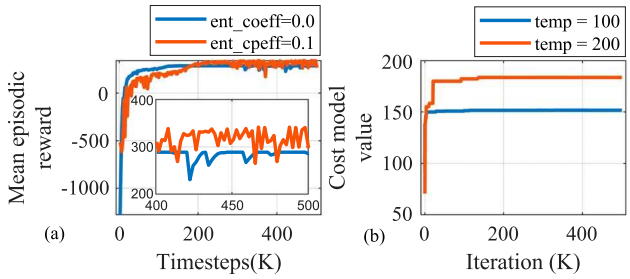

Fig. 8. (a) Impact of entropy coefficient in RL convergence and (b) impact of temperature on SA convergence. Inset shows the zoomed-in version of each plot.

TABLE IV
PPO HYPER-PARAMETERS & THEIR VALUES

n_steps	2048	n_epoch	10
batch_size	64	learning rate	0.0003
clip range	0.2	value func. coef.	0.5
entropy_coeff.	0.1	discount factor	0.99
bias-variance trade-off factor			0.95

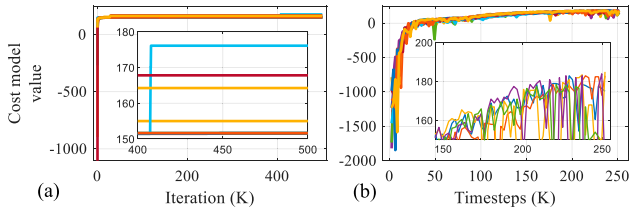

Fig. 9. Convergence behavior of (a) SA and (b) RL for multiple runs with 10 different seed values for case (i) (i.e., 64 chiplets). Inset shows the zoomed-in version of each plot.

higher reward value, albeit with a slightly less stable trajectory. In this case, we use an entropy coefficient of 0.1 to reach higher convergence value. Other significant hyperparameters of PPO algorithm are shown in Table IV.

2) Simulated Annealing: We employ Algorithm 2, initializing it with a randomly chosen candidate solution from the design space. Like PPO, SA's performance is also sensitive to initial temperature, a measure of exploration vs exploitation. As shown is Fig. 8(b), SA achieves significant higher cost model value with higher temperature value. Higher temperature value ensures more exploration by increasing the probability of accepting a worse trial point. As a result, the initial temperature to 200, and a step size of 10 is employed for locating the neighboring points. We do not use the general Metropolis acceptance criterion, $metropolis = exp - \{(O_{curr} - O_{cand})/t\}$, due to the potential for $(O_{curr} - O_{cand})$ to become very large or very small, leading to the $metropolis$ evaluating to either infinity or 0. Instead, we solely utilize the parameter t to statistically accept poorer solutions in the early stages, facilitating exploration of the search space. O_{curr} = cost model value for current design point and O_{cand} = cost model value at candidate design point.

C. Results

1) Performance and Runtime Analysis of Optimizer: In our investigation of the design space, we consider two distinct

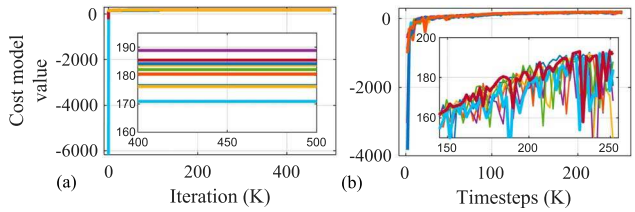

Fig. 10. Convergence behavior of (a) SA and (b) RL for multiple runs with 10 different seed values for case (ii) (i.e., 128 chiplets). Inset shows the zoomed-in version of each plot.

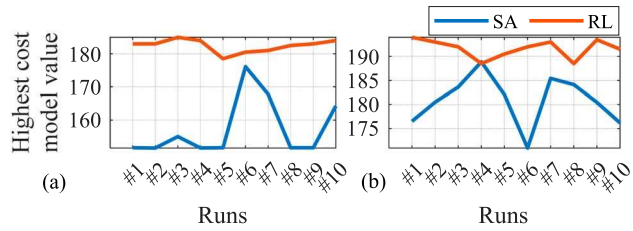

Fig. 11. Highest cost model value achieved by the SA and RL algorithms for multiple runs: (a) for 64 chiplets and (b) for 128 chiplets.

scenarios: case (i), wherein the upper limit for the number of AI chiplets is set to 64, and case (ii), where this upper limit is increased to 128. We ran each of the algorithms multiple times for each cases with different seed values to ensure their convergence stability. Fig. 9 and 10(a) and (b) show the convergence behavior of SA and PPO algorithm for case (i) and case (ii) for 10 runs, respectively. As expected, both algorithms achieve a better cost model value for case (ii) because of its higher throughput, however, due to large packaging cost, case (i) 64 chiplets as the upper bound, is considered more practical. Fig. 11(a) and (b) show the highest cost model value achieved by SA and RL algorithm over 10 runs for case (i) and (ii), respectively. We observe that RL achieves higher cost model values each run and more stable over multiple runs ranging from 178 - 185 for case (i) and 188 - 194 for case (ii). Where SA achieves 151 - 176 and 170 - 188 for case (i) and case (ii), respectively.

The run time of SA for 500K iterations is less than a minute and the run time to train the PPO agent for 250K timesteps is $<$20 mins. We finally integrated several trained RL agents and performed SA optimization on-the-go, and performed an exhaustive search among those SA and RL agents. The final optimizer with 20 SAs and 20 RL trained RL agents take around 10 mins to report the optimized parameter. As the RL is used in inference mode, here the SA dominates the runtime.

2) Optimized Architecture Evaluation: Table V shows the optimized parameter found by the optimizer for both cases for a specific α, β, γ value (user-defined weights on the objective function as explained in Eqn. 17). We observe that the RL PPO algorithm found the best parameter. Please note that multiple design configurations may coexist, achieving almost identical cost model value.

The optimal design point for case (i) consists of 30 3D AI chiplet pairs arranged in a mesh topology 5×6, resulting in 60 chiplets in total. 2 chiplets (forming a pair) are connected with SoIC 3D integration technology with a data rate of 42Gbps

TABLE V
OPTIMIZED PARAMETERS FOR $\alpha, \beta, \gamma = [1, 1, 0.1]$ FOUND BY PPO ALGORITHM

Parameter	Case (i): 64 chiplets as upper bound	Case (ii): 128 chiplets as upper bound
Architecture type	5.5D-Logic-on-Logic	5.5D-Logic-on-Logic
No. of chiplets	60 (30 3D chiplet pairs arranged in 5X6 2.5D mesh)	112 (56 3D chiplet pairs arranged in 7X8 2.5D mesh
Package area; per-chiplet area	900 mm^2; 26mm^2	900 mm^2; 14mm^2
HBMs placement & capacity	4 16GB HBM chiplets @ top, bottom, right, and middle of 5X6 chiplet pairs with a total capacity of 64GB	4 16GB HBM chiplets @ left, right, bottom, and middle of 7X8 chiplet pairs with a total capacity of 64GB
AI2AI 2.5D interconnect type; data rate; link density; trace length	EMIB; 20Gbps; 3100; 1mm	EMIB; 20Gbps; 1450; 1mm
AI2AI 3D interconnect type; data rate; link density	SoIC; 42Gbps; 3200	FOVEROS; 34 Gbps; 4400
AI2HBM 2.5D interconnect type; data rate; link density; trace length	EMIB; 20Gbps; 4900; 1mm	EMIB; 20 GBps; 3850; 1mm

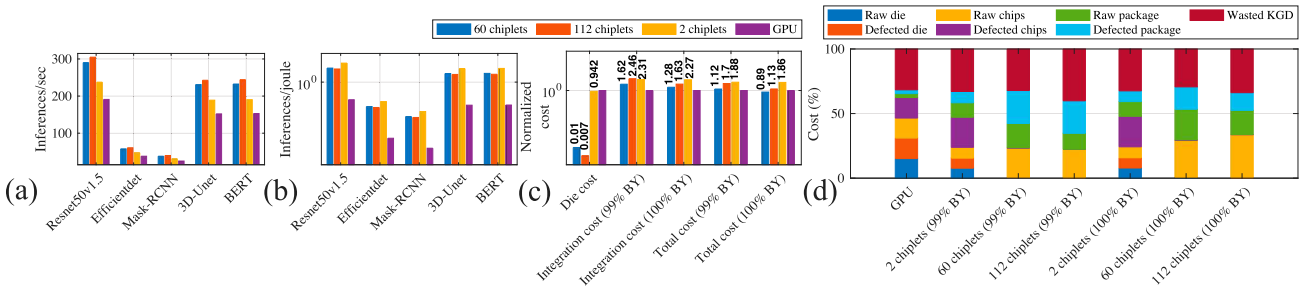

Fig. 12. Comparison of 60-chiplet, 112-chiplet, 2-chiplet and monolithic system: (a) Inferences/sec, (b) Inferences/joule for MLPerf benchmark, and (c) cost. (d) Cost breakdown of monolithic, 2-chiplet, 60-chiplet, and 112-chiplet system at 99% and 100% package bonding yield (BY = bonding yield).

per link and link count of 3200 providing up to 131.25 Tbps of bandwidth. Each chiplet pair is connected with other chiplet pair with 2.5D EMIB integration with a data rate of 20Gbps and a link count of 3100 delivering up to 60 Tbps of bandwidth. Four 16GB HBM chiplets, located at top, right, bottom, and middle of the 5×6 mesh topology, are connected to 2 to 4 neighboring AI chiplets with EMIB 2.5D integration technology with a data rate of 20Gbps per link and a link count of 4900, resulting in a bandwidth of 95 Tbps. The trace length for each 2.5D interconnect is selected as the minimum trace length possible (minimum chiplet-to-chiplet distance). In case (ii), when we increase the maximum number of chiplets to 128, we observe that the optimum design configuration contains 112 chiplets (56 chiplet pairs) and the communication bandwidth decreases for all cases. This is because, as the number of chiplets increases, area per chiplet decreases, resulting in smaller throughput per chiplet, less bandwidth demand, and high system utilization. We observe that 3D architecture, even with area penalty for TSV and TSV-associated keep-out zone [39], achieves $1.52\times$ more logic density than its 2D/2.5D counterpart at the same package size.

We synthesize the chiplet module, found by the optimizer, with Synopsys Fusion Compiler using their 14nm PDK [40] at 1GHz clock frequency and obtain the peak throughput per chiplet, $(ops/sec)_{AI\_chip}$, and energy consumption per MAC operation, E_{op*}. We use these values in our analytical model to estimate the throughput and energy efficiency of the 60 and 112 chiplet system. For cost estimation, we use the model from [6].

Fig. 12 compares the 60-chiplet, 112-chiplet, 2-chiplet and monolithic GPU for MLPerf benchmark [41]. The benchmark features are briefly summarized in Table VI. We observe that 3D

TABLE VI
DNN BENCHMARK FEATURES

Benchmark model	Domain	Dataset	Ops. per forward pass
Resnet50	Image classification	Imagenet	4 GFLOPs
Efficientdet	Light weight object detection	COCO 2017	410 GFLOPs
mask-RCNN	Heavy weight object detection	COCO 2014	447 GFLOPs
3D-UNet	Biomedical image segmentation	KiTS19	947 GFLOPs
BERT	Natural Language Processing	Wikipedia 2020	32 GFLOPs

112-chiplet, 60-chiplet, and 2-chiplet systems achieve $1.60\times$, $1.52\times$, and $1.24\times$ higher throughput of the monolithic one, respectively (Fig. 12(a)). The higher throughput of the chiplet-based system can be explained with three facts. First, higher logic density in 3D logic-on-logic systems in the same area footprint increases the peak theoretical throughput. This explains why all chiplet-based systems show higher throughput than the monolithic one. Second, as the peak theoretical throughput per chiplet (or chiplet pair) increases, the required inter-chiplet communication bandwidth increases. If the active bandwidth cannot sustain the required inter-chiplet (both AI2AI and AI2HBM) bandwidth, system utilization decreases, resulting in decreased achieved throughput. This explains why the 112-chiplet system has the highest throughput compared to the 60-chiplet and 2-chiplet systems. The per-chiplet throughput in the 112-chiplet system is smaller, requiring less inter-chiplet bandwidth and ensuring higher system utilization. Conversely, the 2-chiplet system requires a higher inter-chiplet bandwidth due to its higher per-chiplet throughput, leading to underutilization and reduced overall throughput. Third, as the

number of AI chiplet increases, the inter-chiplet communication latency increases. However, the lower bandwidth penalty of the 112-chiplet system outweighs the higher latency penalty, resulting in a superior overall throughput compared to the 60-chiplet and 2-chiplet systems.

The 2-chiplet, 60-chiplet, and 112-chiplet systems are $4.63\times$, $3.76\times$, and $3.62\times$ energy-efficient (inverse of energy consumption) compared to the monolithic, respectively (Fig. 12(b)). The monolithic system is less energy-efficient than the 3D chiplet based system at iso-throughput. Because, to achieve equal throughput, more than one monolithic chips need to be connected off-board on the PCB, consuming at least one order of magnitude more energy [4] than on-package communication. Among the 3 chiplet-based configurations, 2-chiplet system achieves slightly higher energy-efficiency, $1.23\times$ and $1.28\times$ compared to 60-chiplet and 112-chiplet system, respectively, as it requires less inter-chiplet communication. However, handling the thermal hotspot and heat removal for such large and high-throughput 3D stacked chiplets presents a significant challenge.

Fig. 12(c) shows the cost comparison of the monolithic vs chiplet based systems at different bonding yields. The raw die costs of 60-chiplet, 112-chiplet, and 2-chiplet configuration are $0.01\times$, $0.007\times$, and $0.94\times$, respectively, of the monolithic system. This significant cost difference arises from the low yield (48%) of the monolithic chip of 826mm^2, compared to the 97% and 98% die yield of the 60 and 112 chiplet systems, with a die size of 26mm^2 and 14mm^2, respectively, at 7nm node. In addition to that, the cost of Known Good Dies (KGD) is inversely proportional to the number of KGD (N_{KGD}). As the die area (A) increases, the number of good dies (N_{KGD}) decreases, leading to a substantial increase in cost. The relationship between the cost and die area can be approximated as $cost_{KGD} \propto A^{\frac{5}{2}}$ (taking up to 2 terms of Taylor series expansion of die yield) [4], [6].

We estimate the packaging cost of chiplets at 99% and 100% inter-chiplet bonding yield. With better process control and TSV/pad repair techniques, TSMC reported that the bonding yield can reach 100% [24], [42]. Although the raw die cost is smaller, for chiplet based configurations, the integration cost, including all the defected and wasted chips and packages, of chiplet based system are $1.62\times$ (for 60-chiplet), $2.46\times$ (for 112-chiplet), and $2.31\times$ (for 2-chiplet) higher than the monolithic system at 99% bonding yield. The integration cost improves with the 100% bond yield. Finally, combining the die and integration cost, we observe that the total cost for the 60-chiplet system can achieve the $0.89\times$ cost of the monolithic system with the 100% bonding yield, while the total cost for 112-chiplet configurations is slightly higher ($1.13\times$) than that of the monolithic system. The 2-chiplet system is the most cost inefficient, as it does not benefit from the lower raw die cost and also suffers the high 3D integration cost. Fig. 12(d) shows the breakdown of the total cost of the different configurations. For all configurations, wasted KGD (i.e., wasted chips) consumes a significant amount of total cost, with a maximum of 40% of the total cost for the 112-chiplet system at 99% bonding yield and a minimum of 29% of the total cost for the 60-chiplet at 100% bonding yield. In monolithic and 2-chip systems, die and chip costs (raw and defective) contribute equally to the total cost, and package-related costs (raw and defective) only consume 6% and 20% of the total cost, respectively. On the other hand, in 60 and 112 chiplet-based configurations, the cost of the defected dies and chips are less than 1%. The cost of raw chips and packages (raw and defected) dominates the total cost in these cases. We implement our chiplet in synopsys 14nm free PDK. However, we estimate the cost for 7nm to have a fair comparison between the monolithic one, which was fabricated in 7nm technode [43].

VI. Related Works

A. Chiplet-Based Architecture Exploration

1) DNN Accelerator: SIMBA [13] is a pioneering work in chiplet-based AI accelerator, that integrates 36 NVDLA-like accelerator chiplets on a package. Centaur [51] integrates CPU and FPGA chiplets on package, specially for recommendation system workload. SPRINT [15] is a 64-chiplet system with photonic interconnect for DNN inference. There have been few works in chiplet based architecture focusing on different aspect of design space exploration. NN-Baton [14] proposes a framework for DNN workload mapping and chiplet granularity in small scale (1 to 8 chiplets), however, they do not consider the packaging integration aspect and fabrication cost. While Monad [9] incorporates mapping, resource allocation, communication and different package substrate to optimize for PPA and fabrication cost, their packaging integration design space is limited to 2.5D, excluding 3D. [5] proposes ChipletCloud for LLM inference, however, their chiplets are connected in board-level instead of package level.

2) General Purpose: Some works focus on the exploration of Network-on-Package (NoP) and reliable routing protocols [10] for chiplet-based architecture. [11] explores network topology and cost-aware chiplet placement for 2.5D architecture. [6] puts forward a cost model for evaluating the 2.5D manufacturing cost. [4], [52] suggest the importance of chiplet design space exploration for performance, energy, cost, reliability enhancement. Table VII shows the comparative summary of existing AI accelerator simulator frameworks for design space exploration.

B. RL in Design-Space Exploration

Deep Reinforcement Learning has gained popularity in exploring the design space exploration and optimization of the EDA domain, spanning from front-end (i.e., planning and architectural exploration) [2], [12], [49], and [53] to back-end (i.e., implementation, physical design and circuit design) [54], [55], and [56]. To the best of our knowledge, this work is the first to perform a comprehensive design space search, encompassing resource allocation, placement, packaging architectures (both 2.5D and 3D), and their configurations to optimize for Power, Performance, Area, and Cost (PPAC) using Deep Reinforcement Learning (DRL).

VII. Limitations and Future Works

In order to keep the design space concise and tractable, we limit it as mentioned in Table I and make severalassumptions

TABLE VII
COMPARATIVE SUMMARY OF AI ACCELERATOR SIMULATOR FRAMEWORKS FOR DESIGN SPACE ANALYSIS

	Simulator/Optimizer/ Architecture	Design space/Architectural details	Monolithic/ Chiplet-based
Scale-sim [44]	Performance **Simulator**	(i) Dataflow: WS, IS, OS; (ii) HW resource: No. of PE, on-chip memory size; (iii) Architecture type: Eyeriss, TPU	Monolithic
Timeloop+ accelergy [45]	Mapping **optimizer** + Performance, energy, and area **simulator**	(i) Dataflow: WS, IS, OS; (ii) HW resource: No. of PE, on-chip memory size; (iii) Different memory hierarchies; (iv) Architecture type: Eyeriss, Simba	Monolithic
Maestro [46]	Performance and energy **Simulator +Optimizer**	(i) Dataflow: flexible; (ii) HW resources: Number of PEs, NoC BW/Latency, on-chip memory size; (iii) Architecture type: NVDLA-like	Monolithic
Astra-sim [47]	Performance **Simulator**	(i) Distributed training: data, model, hybrid parallelism; (ii) Hierarchical collective algorithms: on-load, off-load; (iii) Fabric design: number of links & latency/BW per link; (iv) Fabric topology: pt-to-pt, 2D/3D Torus	Multi-chip (package/board level)
STONNE [48]	Performance, energy and area **Simulator**	Architecture type: flexible & reconfigurable architecture	Monolithic
Confucuix [49]	Performance and energy **optimizer**	(i) HW resource: Number of PE, on-chip buffer size	Monolithic
SIMBA [13]	Performance & power **Optimizer** + 36-chiplet **Architecture**	(i) Mapping and tiling strategies; (ii) 36 NVDLA chiplets connected in 2D mesh	Multi-chip (package level)
SPRINT [15]	64-chiplet **Architecure**	64-chiplet architecture with photonic interconnect for interchiplet communication	Multi-chip (package level)
NN-Baton [14]	**Simulator** + **Optimizer**	(i) No. of accelerator chiplets: 1, 2, 4, 8; (ii) On-chip memory: 36 to 642KB; (iii) Different mapping strategies; (iv) Routing topology: Ring	Multi-chip (package level)
Moand [9]	**Optimizer**	(i) No. of accelerator chiplets; (ii) mapping & tiling; (iii) packaging; (iv) network topology; (v) placement	Multi-chip (package level)
TVLSI'20 [50]	**Simulator**	(i) 64-core ROCKET-64 architecture; (ii) 2.5D interposer-based centralized NoC	Multi-chip (package level)
Chiplet-gym (This work)	Performance, power, area and cost **Optimizer**	(i) No. of AI accelerator chiplets, no. & location of HBM chiplets; (ii) package architecture: 2.5D, 3D; (iii) Interconnect types & configuration (details in Table I)	Multi-chip (package level)

as mentioned in Section V-A. We also assume that the HBM3e chiplets have their integrated memory controller and NoC router that can be used as a node in the mesh topology. The analysis of the cost of additional chips with the NoC+memory interface, exploring other routing topology such as p2p with photonic interconnects, H tree, bus, ring etc., exploring more heterogeneous architectures, multi-tier 3D-stacks, placement of host CPU chiplets and exploring their different layouts are future works.

VIII. CONCLUSION

This paper proposes Chiplet-Gym to explore the design space of chiplet-based AI accelerators to optimize for Power, Performance, Area, and Cost (PPAC). To evaluate the design points, we analytically model the PPAC for chiplet-based AI accelerator. With reinforcement learning and simulated annealing, the optimizer is robust and efficient in locating the global or near-global optima of the design space for PPAC. The results show that the optimizer finds the design point that achieves 1.52× throughput, 0.27× energy, and 0.89× cost of its monolithic counterpart in iso-area.

REFERENCES

[1] Y. Chang et al., "A survey on evaluation of large language models," *ACM Trans. Intell. Syst. Technol.*, vol. 15, no. 3, pp. 1–45, 2024.

[2] J. You et al., "Zeus: Understanding and optimizing GPU energy consumption of DNN training," in *Proc. 20th USENIX Symp. Netw. Syst. Des. Implementation (NSDI)*, 2023.

[3] W. Chen and W. R. Bottoms, "Heterogeneous integration Roadmap," in *Proc. Int. Conf. Electron. Packaging (ICEP)*, 2017, pp. 302–305. [Online]. Available: https://eps.ieee.org/technology/heterogeneous-integration-roadmap/2021-edition.html

[4] A Sangiovanni-Vincentelli et al., "Automated design of chiplets," in *Proc. Int. Symp. Phys. Des.*, 2023.

[5] H. Peng, et al., "Chiplet Cloud: Building AI supercomputers for serving large generative language models," *arXiv:2307.02666*, 2023.

[6] Y. Feng and K. Ma, "Chiplet actuary: A quantitative cost model and multi-chiplet architecture exploration," in *Proc. 59th ACM/IEEE Des. Automat. Conf.*, 2022, pp. 121–126.

[7] H. Jiang, "Intel's Ponte Vecchio GPU : Architecture, systems & software," in *Proc. IEEE Hot Chips 34 Symp.*, 2022, pp. 1–29.

[8] S Naffziger et al., "Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: Industrial product," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit.*, 2021, pp. 57–70.

[9] X. Hao, Z. Ding, J. Yin, Y. Wang, and Y. Liang, "Monad: Towards cost-effective specialization for chiplet-based spatial accelerators," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des. (ICCAD)*, 2023, pp. 1–9.

[10] T. Wang, F. Feng, S. Xiang, Q. Li, and J. Xia, "Application defined on-chip networks for heterogeneous chiplets: An implementation perspective," in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, 2022, pp. 1198–1210.

[11] A. Coskun et al., "Cross-layer co-optimization of network design and chiplet placement in 2.5-D systems," in *Proc. IEEE Trans. Comput.-AIDED Des. Integr. Circuits Syst.*, vol. 39, no. 12, pp. 5183–5196, Dec. 2020.

[12] T. -R. Lin, D. Penney, M. Pedram, and L. Chen, "A deep reinforcement learning framework for architectural exploration: A routerless NoC case study," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, 2020, pp. 99–110.

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

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

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

[16] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, pp. 6000–6010, 2017.

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

[18] N Jouppi et al., "TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings," in *Proc. 50th Annu. Int. Symp. Comput. Archit.*, 2023, pp. 1–14.
[19] K. Mishty and M. Sadi, "System and design technology co-optimization of SOT-MRAM for high-performance AI accelerator memory system," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 43, no. 4, pp. 1065–1078, Apr. 2024.
[20] K. C.-H. Hsieh, "Designs of communication circuits for side-by-side and stacked chiplets," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2021.
[21] M. Ravi et al., "Embedded Multi-die Interconnect Bridge (EMIB) – A High Density, High Bandwidth Packaging Interconnect," in *Proc. IEEE 66th Electron. Compon. Technol. Conf.*, 2016.
[22] R. Mathur, A. K. A. Kumar, L. John, and J. P. Kulkarni, "Thermal-aware design space exploration of 3-D systolic ML accelerators," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 7, no. 1, pp. 70–78, Jun. 2021.
[23] D. B. Ingerly et al., "Foveros: 3D integration and the use of face-to-face chip stacking for logic devices," in *Proc. IEEE Int. Electron Devices Meeting*, 2019, pp. 19.6.1–19.6.4.
[24] F.-C. Chen, W.-C. Chiou, and D. C. H. Yu, "System on integrated chips (SoICTM) for 3D heterogeneous integration," in *Proc. IEEE 69th Electron. Compon. Technol. Conf.*, 2019, pp. 594–599.
[25] E. J. Marinissen, T. McLaurin, and H. Jiao, "IEEE Std P1838: DfT standard-under-development for 2.5D-, 3D-, and 5.5D-SICs," in *Proc. 21th IEEE Eur. Test Symp. (ETS)*, 2016, pp. 1–10.
[26] V Sze et al., "How to evaluate deep neural network processors: TOPS/W (alone) considered harmful," *IEEE Solid-State Circuits Mag.*, vol. 12, no. 3, pp. 28–41, Aug. 2020.
[27] S Bharadwaj et al., "Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling," in *Proc. 57th ACM/IEEE Des. Automat. Conf. (DAC)*, 2020, pp. 1–6.
[28] K. Mishty and M. Sadi, "System and design technology co-optimization of chiplet-based AI accelerator with machine learning," in *Proc. Great Lakes Symp. VLSI*, 2023, pp. 697–702.
[29] "HBM3." Mar. 2022. Accessed: Jan. 2024. [Online]. Available: https://www.jedec.org/
[30] "AMD MI300." Mar. 2023. Accessed: Jan. 2024. [Online]. Available: https://www.amd.com/en.html
[31] T. Tang and Y. Xie, "Cost-aware exploration for chiplet-based architecture with advanced packaging technologies," 2022, *arXiv:2206.07308*.
[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.
[33] "Openai gym." Mar. 2022. Accessed: Jan. 2024. [Online]. Available: https://www.gymlibrary.dev/index.html
[34] S. John et al., "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.
[35] "Stable-baselines3." Mar. 2021. Accessed: Jan. 2024. [Online]. Available: https://stable-baselines3.readthedocs.io/en/master/index.html
[36] H. T. Kung et al., "Systolic building block for logic-on-logic 3D-IC implementations of convolutional neural networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2019, pp. 1–5.
[37] P. Shukla, D. Aguren, T. Burd, A. K. Coskun, and J. Kalamatianos, "Temperature-aware sizing of multi-chip module accelerators for multi-DNN workloads," in *Proc. Des., Automat. & Test Europe Conf. & Exhib. (DATE)*, 2023, pp. 1–6.
[38] 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.
[39] JM Joseph et al., "Architecture, dataflow and physical design implications of 3D-ICs for DNN-accelerators," in *Proc. 22nd Int. Symp. Qual. Electron. Des.,* Piscataway, NJ, USA: IEEE Press, 2021, pp. 60–66.
[40] "Synopsys 14nm PDK." Accessed: Jan. 2024. [Online]. Available: https://www.synopsys.com/academic-research/university.html
[41] "MLPerf Benchmark." Mar. 2024. Accessed: Jan. 2024. [Online]. Available: https://www.nvidia.com/en-us/data-center/resources/mlperf-benchmarks/
[42] Q. X. L. Jiang and B. Eklow, "On effective through-silicon via repair for 3-D-stacked ICS," in *Proc. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, Apr. 2013, pp. 559–571.
[43] "NVIDIA Ampere100 GPU." Accessed: Jan. 2024. [Online]. Available: https://www.nvidia.com/en-us/data-center/ampere-architecture/
[44] A. Samajdar, J. M. Joseph, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "A systematic methodology for characterizing scalability of DNN accelerators using SCALE-Sim," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2020, pp. 58–68.
[45] P. Angshuman et al., "Timeloop: A systematic approach to DNN accelerator evaluation," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw.*, 2019, pp. 304–315.
[46] K. Hyoukjun, P. Chatarasi, V. Sarkar, T. Krishna, M. Pellauer and A. Parashar, "MAESTRO: A data-centric approach to understand reuse, performance, and hardware cost of DNN mappings," *IEEE Micro*, vol. 40, no. 3, pp. 20–29, May/Jun. 2020.
[47] W. William et.al., "ASTRA-sim2.0: Modeling hierarchical networks and disaggregated systems for large-model training at scale," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2023, pp. 283–294.
[48] F. Muñoz-Martínez, J. L. Abellán, M. E. Acacio, and T. Krishna, "STONNE: Enabling cycle-level microarchitectural simulation for DNN inference accelerators," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, 2021, pp. 201–213.
[49] S.-C. Kao, G. Jeong, and T. Krishna, "Confuciux: Autonomous hardware resource assignment for DNN accelerators using reinforcement learning," in *Proc. 53rd Annu. IEEE/ACM Int. Symp. Microarchit. (MICRO),* Piscataway, NJ, USA: IEEE Press, 2020, pp. 622–636.
[50] J. Kim et al., "Architecture, chip, and package codesign flow for interposer-based 2.5-D chiplet integration enabling heterogeneous IP reuse," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 11, pp. 2424–2437, Nov. 2020.
[51] R. Hwang, T. Kim, Y. Kwon and M. Rhu, "Centaur: A chiplet-based, hybrid sparse-dense accelerator for personalized recommendations," in *Proc. ACM/IEEE 47th Annu. Int. Symp. Comput. Archit.*, Los Alamitos, CA, USA, Jun. 2020, pp. 968–981.
[52] G. H. Loh and R. Swaminathan, "The next era for chiplet innovation," in *Proc. Des., Automat. & Test Europe Conf. & Exhib. (DATE)*, 2023, pp. 1–6.
[53] S Krishnan et al., "Multi-agent reinforcement learning for microprocessor design space exploration," 2022, *arXiv:2211.16385*.
[54] A Mirhoseini et al., "Chip placement with deep reinforcement learning," 2020, *arXiv:2004.10746*.
[55] H. Wang et al., "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. 57th ACM/IEEE Des. Automat. Conf.,* 2020, pp. 1–6,
[56] K. Settaluri, A. Haj-Ali, Q. Huang, K. Hakhamaneshi, and B. Nikolic, "AutoCkt: Deep reinforcement learning of analog circuit designs," in *Proc. Des., Automat. & Test Europe Conf. & Exhib. (DATE),* Piscataway, NJ, USA: IEEE Press, 2020, pp. 490–495.

Kaniz Mishty received the B.S. degree in electronics and communication engineering from Khulna University of Engineering and Technology, Bangladesh, in 2018. She is currently working toward the Ph.D. degree in electrical and computer engineering with Auburn University, AL, USA. She interned with Apple Inc. in 2022, 2023 and Qualcomm in 2021 on AI application in System-on-Chip and custom circuit design. Her research interests include energy efficient AI hardware design and AI/ML in CAD.

Mehdi Sadi (Member, IEEE) received the B.S. degree from Bangladesh University of Engineering and Technology, in 2010, the M.S. degree from the University of California at Riverside, USA, in 2011, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, USA, in 2017. He is an Assistant Professor with the Department of Electrical and Computer Engineering at Auburn University, Auburn, AL. He was a Senior R&D System-on-Chip Design Engineer in the Xeon server CPU design team at Intel Corporation in Oregon. He has published more than 25 peer-reviewed research papers. He was the recipient of Semiconductor Research Corporation best in session award and Intel Xeon Design Group recognition awards. His current research interests include developing algorithms and CAD techniques for implementation, design, reliability, and security of AI, and brain-inspired computing hardware. His research also spans into developing machine learning/AI enabled design flows for system-on-chip, and design-for-robustness for safety-critical AI hardware systems.