System and Design Technology Co-optimization of Chiplet-based AI Accelerator with Machine Learning

Kaniz Mishty
Auburn University
Auburn, AL, USA
kzm0114@auburn.edu

Mehdi Sadi
Auburn University
Auburn, AL, USA
mzs0190@auburn.edu

ABSTRACT

With the availability of advanced packaging technology and its attractive features, the chiplet-based architecture has gained traction among chip designers. The large design space and the lack of system and package-level co-design methods make it difficult for the designers to create the optimum design choices. In this research, considering the colossal design space of advanced packaging technologies, resource allocation, and chiplet placement, we design an optimizer that looks for the design choices that maximize the Power, Performance, and Area (PPA) and minimize the cost of the chiplet-based AI accelerator. Inspired by the Bayesian approach for black-box function optimization, our optimizer guides the search space toward global maxima instead of randomly traversing through the search space. We analytically synthesize a dataset from the search space and train an ML model to predict the target value of our defined cost function at the optimizer-suggested points. The optimizer locates the optimum design choices from the specified search space (≥1M data points) with minimal iterations (≤ 200 iterations) and trivial run time.

CCS CONCEPTS

• **Computer systems organization** → Multichip architectures; Heterogeneous integration; Interconnection architectures; Deep Learning Hardware.

KEYWORDS

Chiplet-based architectures, Deep Learning hardware, advanced packaging, 2.5D, 3D, PPA optimization, STCO

ACM Reference Format:
Kaniz Mishty and Mehdi Sadi. 2023. System and Design Technology Co-optimization of Chiplet-based AI Accelerator with Machine Learning. In *Proceedings of the Great Lakes Symposium on VLSI 2023 (GLSVLSI '23), June 5–7, 2023, Knoxville, TN, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3583781.3590233

1 INTRODUCTION

In the era of memory and compute-hungry Big Data and Artificial Intelligence (AI), the stagnation of Moore's law and Dennerd's scaling, and die size reaching the reticle limit has triggered the chip design industry to transition from the monolithic IC to heterogeneous chiplet-based architecture. With the advent of advanced packaging technologies and interconnects, the chiplet-based heterogeneous integration has opened up a new dimension of chip design, "More-than-Moore" [2]. In chiplet-based system, multiple chiplets (i.e., SoCs) of diverse functionalities (e.g., logic dies, memories, analog IPs, SerDes, accelerator 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. [2].

The value proposition of chiplet-based architectures is manifold. First, compared to multiple monolithic SoCs interconnected via off-package or off-board links such as PCIe, NVLink, CXL etc. [2], package-level integration of multiple monolithic SoCs via 2.5D or 3D has accelerated performance and lower energy consumption. Training any state-of-the-art AI or Deep Learning model with a single GPU is nearly impossible due to extreme computing and memory demands. The data centers are equipped with clusters of powerful computers connected via PCIe. Even though these supercomputers can deal with large workloads, off-board communications come with a great expense of power, performance, and area. Combining as many chiplets as possible at the package level will alleviate the off-package communication costs. Second, it enables IP reuse and provides flexibility in picking the best process node for the required IP. For example, SerDes IO, analog, and RF IPs have higher design complexity, lower shrink factor, and lower performance or power gain in advanced tech nodes compared to digital logic IPs [7] [19]. Heterogeneous chiplet architecture enables the integration of analog and IO blocks from mature technology whereas digital logic blocks from the latest tech node ensure leading performance, performance/$, and performance/W at the shortest time-to-market [2] [19]. Third, it yields better due to smaller die sizes [2]. Finally, it enables a shorter IC design cycle, lower development, manufacturing, and NRE cost by reusing pre-existing chiplets [2]. Leveraging the chiplet concept, in 2017 the DARPA announced the Common Heterogenous Integration and Intellectual Property (IP) Reuse Strategies program (which is still going on) to achieve 3 goals: (i) Extending Moore's Laws, (ii) Enabling heterogeneous integration, and (iii) Empowering system integrators, to cope with the ever explosive growth of the powerful semiconductor needs [2].

Few commercial chiplet-based products are available in the market [6] [10] [19] [20] and most are designed and developed at vertically integrated companies [21]. The vertically integrated products do not ensure the versatile chiplet based ecosystem. The complete adoption of chiplet-based architecture is holding off because of the lack of top-level/system-level heterogeneous design aggregation, planning, and optimization. Designers must consider system requirements, stacking/packaging technologies, interconnect architecture, resource management, chiplet granularity, placement, reliability, scalability, etc., to optimize Power, Performance, and Area (PPA) [7] [21]. Currently, many flavors of packaging technologies, both from 2.5D and 3D, are available from the industry leaders,

GLSVLSI '23, June 5–7, 2023, Knoxville, TN, USA

ACM ISBN 979-8-4007-0125-2/23/06…$15.00
https://doi.org/10.1145/3583781.3590233

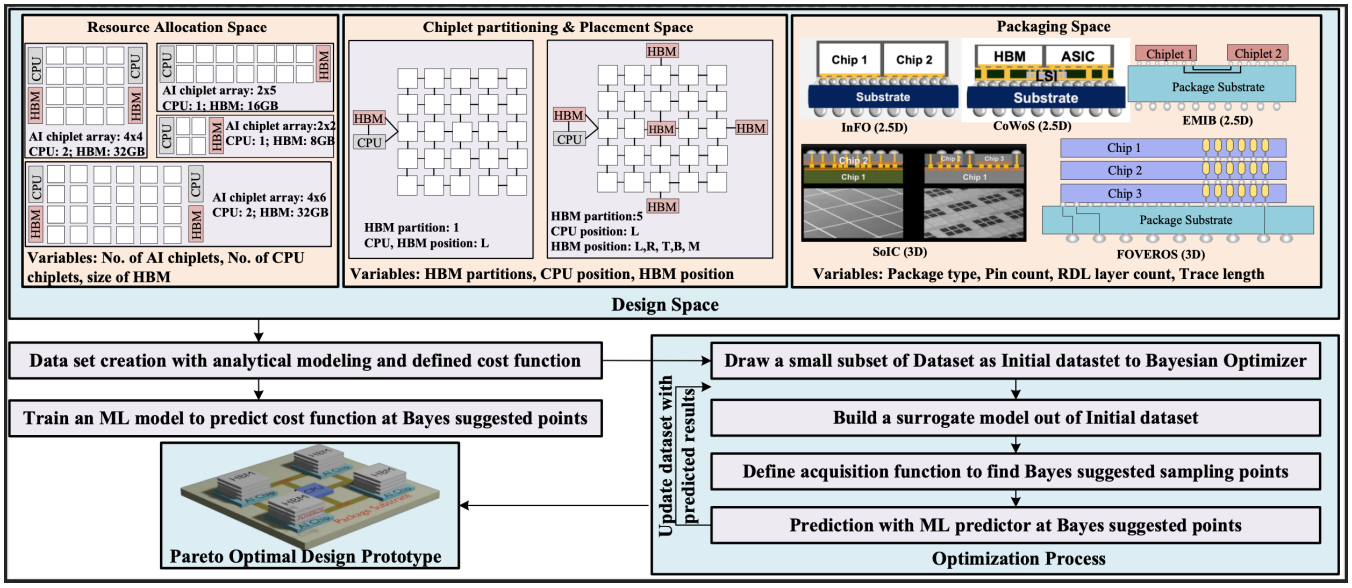

Figure 1: Overview of the proposed optimization scheme. Examples of a few possible design scenarios are shown in the Design space. L, R, T, B, M represent left, right, top, bottom, and middle, respectively.

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 [2]. The various packaging technologies differ in fabrication cost and complexity, performance, and underlying integration technologies [2]. 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 (in terms of time and resources) simulation environment of chip design exacerbates this problem.

In this work, we strive to bridge the gap between the system requirements and design aggregation, planning, and optimization for chiplet-based architecture. We incorporate resource allocation, partitioning and placement of chiplets, and packaging technologies in our optimization problem and solve for the optimum PPA within the cost budget. The contributions of the paper are as follows.

- In our chiplet-based PPA optimization problem, we take resource allocation such as the number of AI chiplets, memory size and bandwidth; partitioning and placement of the chiplets, such as aspect ratio of the accelerator chiplet arrays, layout, and logical placement of accelerator, memory and host CPU; and different packaging technology (i.e., CoWoS, EMIB, InFO, SoIC, and FOVEROS [2]), and their attributes such as bandwidth, bump pitch density, cost and complexity, etc., to integrate the chiplets in a single package into account.
- Due to the expensive and intricate simulation behavior of each step of chip design (e.g., logic synthesis, STA analysis, PnR, EMIR and reliability analysis), we model it as a black box function and solve the problem with *bayesian approach* to look for global maxima. We build an objective function that minimizes *Area*, *Energy*, *Latency*, and *Cost & Complexity* and maximizes *Throughput* and *Bandwidth*.
- Because of the unavailability of public data on chip design that can act as a dataset and expensive simulation methodology, we build an analytical model to create a synthetic dataset. Based on the fundamental concepts of the VLSI design and very few publicly available data from industrial products as base value, our analytical model is very efficient (in terms of time and resources) and alleviates the need for iterative expensive simulations for dataset creation. We also train an ML model on our custom dataset to mimic the simulation with CAD tools during our optimization.

The rest of the article is organized as follows. Section II briefly reviews the literature; section III illustrates the methodology; section IV provides the experimental results; and section V concludes the paper.

2 RELATED WORK

There have been a few explorations of the chiplet-based architecture for DNN accelerator; some of them focus on optimizing the workload mapping to the chiplets [8] [23] [24] , while others focus on the exploration of the Network-on-Packcage (NoP) and reliable routing protocols for chiplets based architecture [25] [26]. With a handful of packaging technologies available for chiplet integration and their different configuration, no studies have, to our knowledge, performed the co-optimization of different packagaing technologies (including 2.5D and 3D), resource allocation, and placement to optimize the system PPA for AI accelerators. [4] performed a cross-layer co-optimization network design and chiplet placement, but only for 2.5D systems. The use of Machine Learning, Reinforcement Learning, and different optimization schemes to optimize different domains of ASIC design, including AI accelerator, are also available in the literature [11] [12] [13] [16] [27] . However, none of them optimized different packaging technology and their configuration.

3 SYSTEM TECHNOLOGY CO-OPTIMIZATION OF CHIPLET-BASED AI ACCELERATOR

The fundamental goal of our optimization problem is to minimize the system area, energy consumption, latency, and cost & complexity while maximizing the system performance and bandwidth based on the system application. Our optimization methodology comprehends a wide range of design spaces of resource allocation, chiplet partitioning, logical placement, and packaging technology to reach an optimum solution. In this section, we explain our optimization methodology in detail. The overview of our optimization scheme is shown in Fig. 1.

3.1 Formulating the Objective function

To minimize the system area, energy consumption, latency, cost & complexity and maximize the system throughput and bandwidth based on system requirements, we define our objective function as

$$Maximize: \alpha * (\frac{1}{A}) + \beta * (Th + BW + \frac{1}{L}) + \gamma * (\frac{1}{E}) + \delta * (\frac{1}{CC}) \quad (1)$$

Where, $A = Area$, $Th = Throughput$, $BW = Bandwidth$, $L = Latency$, $E = Energy/bit\ data\ communication$, $CC = design\ and\ manufacturing\ associated\ Cost\&Complexity$ of the system. α, β, γ, and δ are coefficients that allow the users to put the specific weightage on the specific parameters during the optimization. We define $\sum(\alpha, \beta, \gamma, \delta) = 1$, where any coefficient with 0 value represents no weightage (i.e., that term will be dropped from the objective function) and any coefficient with 1 value represents the highest weightage. The user needs to choose the appropriate values of the coefficients based on the system requirement. For example, the energy coefficient, γ, should be higher for low-power applications. If the coefficients have equal values (i.e., 0.25 each), the optimizer will put equal effort into optimizing all parameters.

3.2 Dataset Generation

This subsection will describe our analytical model to create the dataset and each component of our objective function. Our dataset consists of Resource allocation space, Chip partitioning and chiplet placement space, and packaging space.

3.2.1 Resource Allocation. Resource allocation impacts the device's performance and area. The $Area(A)$ and $Throughput(Th)$ of our objective function come directly from the resource allocation space. The system latency also depends on resource allocation. However, we discuss the latency, in detail, in section 3.2.2 as the chiplet partitioning and placement also impact it. We define the system area A as

$$A = \sum_{i=4, j=1}^{p,q} (acc\_area_i + cpu\_area_j + hbm\_area) + routing\_area \quad (2)$$

Where, $acc\_area = AI\ accelerator\ area$; i, varying from 4 to p, is the number of AI accelerator chiplets integrated on a package; $cpu\_area = area\ of\ each\ CPU$; j is the number of CPUs that can vary from 1 to q; $routing\_area$ is the area dedicated for the chiplet-to-chiplet routing of the signal. $routing\_area$ again depends on the network topology (e.g., mesh, crossbar, torus etc.) and the packaging technology (e.g., 2.5D and 3D). In this work, we consider AI accelerator chiplets connected in a 2D ($X \times Y$) mesh topology to create our dataset. We explore different aspect ratios of the array by varying the X dimension from 2 to m and the Y dimension from 2 to n, where, $p = m * n$ and $m, n \geq 2$. The host CPU and HBM chiplets can be interconnected in 2.5D or 3D stacked on top of the AI chiplets (will be discussed in detail in section 3.2.2). The system Throughput, Th, is defined as

$$Th = \sum_{i=4}^{p} Peak\_Th_i \quad (3)$$

Where $Peak\_Th$ is the peak attainable throughput of AI accelerator chiplets and i is the number of the AI chiplets. We take the area and throughput of each AI chiplet from SIMBA [23], CPU area from [19], and HBM area from [1] and using equations 2-3, we calculate the system area and throughput for various resource amount.

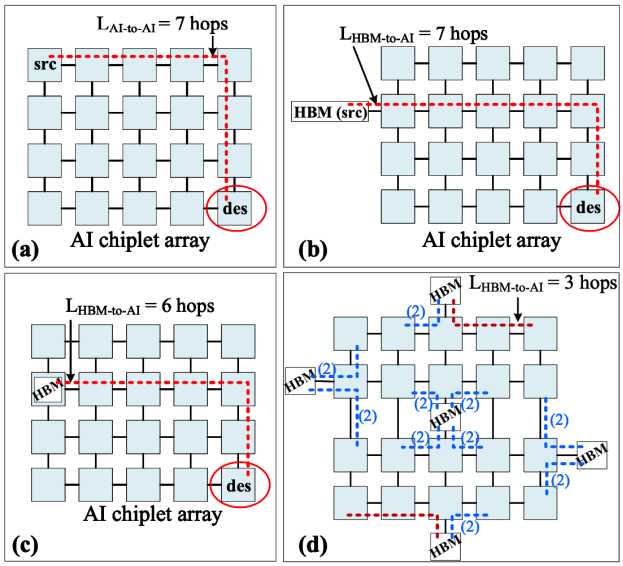

Figure 2: Illustration of latency (in terms of hop) calculation. (a) AI-to-AI chiplet communication, considering the farthest chiplets as source-destination pair. (b) One HBM chiplet, located at the left connected in 2.5D, and the farthest AI chiplet as source-destination pair. (c) One HBM chiplet, 3D-stacked 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.

Table 1: Resource allocation, chiplet partitioning, and placement space

Parameter	Values
AI chiplet X dim	2 to 16 @ step of 1
AI chiplet Y dim	2 to 16 @ step of 1
No. of CPU/HBM chiplets	1 to 5 @step of 1
CPU/HBM locations	[left, right, top, bottom, middle, 3D-stacked]; $2^6 - 1$ locations

3.2.2 Chiplet Partitioning and Placement. Chiplet partitioning and the logical placement of the chiplets have impacts on the device's performance. In this section, we consider dividing the total allocated HBM into multiple chiplets and place the chiplets in multiple positions. The partitioning of a large chunk of memory into multiple memory chiplets (instead of placing the large memory in

Table 2: Analytical expressions for CPU/HBM-to-AI chiplet latency

#CPU/HBM	Locations	$L_{CPU/HBM-to-AI}$ 2.5D	3D
1	Left/right	$m + \lfloor \frac{n}{2} \rfloor$	$m + \lfloor \frac{n}{2} \rfloor - 1$
1	Top/bottom	$\lfloor \frac{m}{2} \rfloor + n$	$\lfloor \frac{m}{2} \rfloor + n - 1$
1	Middle	$\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$	m,n(even):$\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil$ m,n (odd): $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 2$ otherwise: $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$
2	Left+right	$\lceil \frac{m}{2} \rceil + \lfloor \frac{n}{2} \rfloor$	$\lceil \frac{m}{2} \rceil + \lfloor \frac{n}{2} \rfloor - 1$
2	Top+bottom	$\lfloor \frac{m}{2} \rfloor + \lceil \frac{n}{2} \rceil$	$\lfloor \frac{m}{2} \rfloor + \lceil \frac{n}{2} \rceil - 1$
2	Left/right + Top/Bottom	$min(m,n) + \lfloor \frac{max(m,n)}{2} \rfloor$	$min(m,n) + \lfloor \frac{max(m,n)}{2} \rfloor - 1$
2	Left/right/top/bottom + middle	$\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$	m,n (even):$\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil$ m,n (odd): $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 2$ otherwise: $\lceil \frac{m}{2} \rceil + \lceil \frac{n}{2} \rceil - 1$
3	Left+right+middle, Top+bottom+middle	$\lfloor \frac{m}{2} + \frac{n}{2} \rfloor - 1$	m,n (odd):$\lfloor \frac{m}{2} + \frac{n}{2} \rfloor - 2$ otherwise:$\lfloor \frac{m}{2} + \frac{n}{2} \rfloor - 1$
3	Left+right+top/bottom, Top+bottom+left/right	$\lfloor \frac{m}{2} + \frac{n}{2} \rfloor$	$\lfloor \frac{m}{2} + \frac{n}{2} \rfloor - 1$
3	Left/right+top/bottom+middle	m,n (even): $min(m,n) - 1$ otherwise: $min(m,n)$	m,n (odd): $min(m,n) - 1$ otherwise: $min(m,n)$
4	Left+right+top+bottom	m,n (even): $\frac{max(m,n)}{2} + 1$ otherwise: $\lceil \frac{max(m,n)}{2} \rceil$	m,n (even): $\frac{max(m,n)}{2}$ otherwise: $\lceil \frac{max(m,n)}{2} \rceil - 1$
4	Middle+left+right+top/bottom, Middle+top+bottom+left/right	m,n (even): $\frac{max(m,n)}{2}$ otherwise: $\lceil \frac{max(m,n)}{2} \rceil$	m,n (even): $\frac{max(m,n)}{2} + 1$ m,n (odd): $\lceil \frac{max(m,n)}{2} \rceil - 1$ otherwise: $\lceil \frac{max(m,n)}{2} \rceil$
5	Left+right+top+bottom+middle	m,n (even): $\frac{max(m,n)}{2} + 1$ otherwise: $\lceil \frac{max(m,n)}{2} \rceil$	m,n (even): $\frac{max(m,n)}{2}$ otherwise: $\lceil \frac{max(m,n)}{2} \rceil - 1$

one place) and placing these multiple memory chiplets in different positions improves the system latency. Because the communication latency depends on the physical location of the data [18] [23]. Fig. 2 illustrates how chiplet partitioning and placement improve the system latency. Please note this placement only considers logical placement, i.e., chiplet-to-chiplet logical connection. We consider a 2D mesh of AI accelerator chiplets. Therefore, there are 6 locations: left, right, top, bottom, middle, and 3D stacking, to place the host CPUs and HBMs around the AI chiplet array. These locations result in $2^6 - 1$ combinations for CPU and HBM placements. Table 1 presents all parameters and their values for the resource allocation, chiplet partitioning, and placement space.

The chiplet-to-chiplet data communication latency, L, of our objective function is calculated from the parameters of this space. The actual chiplet-to-chiplet data communication latency depends on various factors, such as process technology node, number of repeaters/buffer between source to destination, routing arbitration, thermal conditions, signal referencing circuitry, etc. However, we calculate the latency in terms of data source-destination hop counts. The reason is that running the actual simulation and finding the communication latency is costly and time-consuming. Instead, through our analytical modeling we express the latency in terms of the AI chiplet X and Y dimensions for different locations of CPU and HBM chiplets. While the hop count does not capture the actual latency, the actual latency depends on the hop count, and we achieve a significant run time improvement to estimate the overall system latency. We take the worst-case source-destination pair as the latency to be more conservative.

The L of the objective function comprises AI-to-AI, CPU-to-AI, and HBM-to-AI communication latency.

$$L = L_{AI-to-AI} + L_{CPU-to-AI} + L_{HBM-to-AI} \tag{4}$$

AI-to-AI communication latency is

$$L_{AI-to-AI} = m + n - 2 \tag{5}$$

Where, m = no. of AI chiplets in X dimension, and n = no. of AI chiplets in Y dimension. CPU/HBM-to-AI latency depends on the CPU/HBM position and the AI chiplets array dimension. In table 2, we present the expressions of L for different CPU and HBM locations.

Table 3: Parameters and Values of Packaging Space

Parameter	Values
Package type	CoWoS (2.5D), EMIB(2.5D), InFO(2.5D), FOVEROS(3D), SoIC(3D)
IO density/mm^2	500 to 1,000,000 @ step of 1000
Data rates (Gbps)	10 to 50 @ step of 2
Trace length (mm)	0.5 to 10 @ step of 2
RDL layer counts	1 to 14 @ step of 2

3.2.3 Packaging Space. The packaging space plays the most important role as it impacts half of the parameters of the objective function: *Bandwidth* (BW), *Energy per bit data communication* (E), and *Cost and Complexity* (CC). We explore different types of packaging architecture, their data rates, IO density, trace length (bump to bump distance between two interconnected dies), and RDL counts [3], [9], [14], [15], [17] to calculate the bandwidth, energy/bit, and cost and complexity. The bandwidth is calculated as a function of data rate per pin:

$$BW = P * DR \tag{6}$$

Where, $P = IO/mm^2$, i.e., pin count, and $DR = data\ rate\ /pin$. Taking data form [1] [7], [17], as base value, we interpolate the energy/bit, E, and RDL counts as a function of trace length and bump

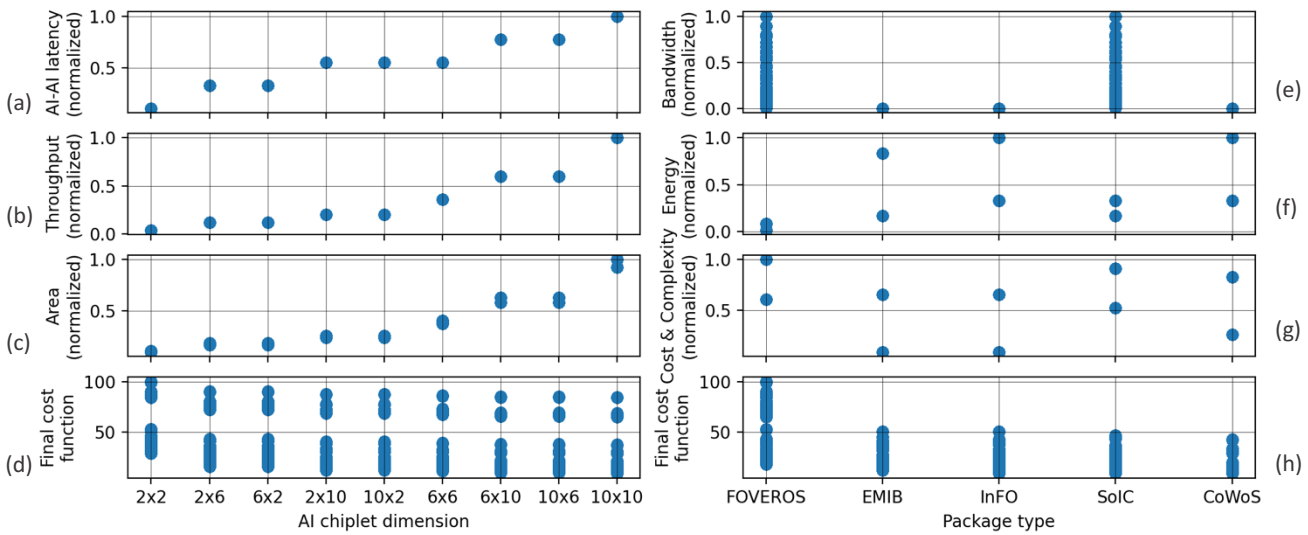

Figure 3: Interdependency between different parameters of the state space. (a-d) AI-AI chiplet communication latency, throughput, area, and final cost function as functions of the number of AI chiplets, respectively. (e-f) Bandwidth, Energy/bit data transfer, cost & complexity, and final cost function as functions of package type, respectively. Except for the final cost functions, the data are normalized w.r.t. the maximum values of the corresponding feature domain.

pitch [7]. We comprehend the heat management, integration cost, TSV-associated overhead (e.g. keep out zone), signal communication integrity to quantify the cost and complexity [3], [9], [14], [15], [17]. The parameters and their values that are considered to create the dataset are shown in Table 3.

3.3 Optimization Scheme

Our optimization engine is built based on the *Bayesian Optimization* that tries to maximize the objective function (eqn. 1). *BayesOpt* comprises two main components: (i) a Bayesian Statistical model, commonly known as the Surrogate model to capture the behavior of the objective function, and (ii) an acquisition function to decide which region to sample next [5].

3.3.1 Surrogate Model. We model our objective function using a GPRegressor [22]. We use Radial Basis Function (RBF) kernel

$$\sum_{0}(x, x') = \alpha_0 exp(-||x - x'||^2) \tag{7}$$

where, $||x-x'||^2 = \sum_{i=1}^{d} \alpha_i (x_i - x_i')^2$, and $\alpha_{0:d}$ are the parameters of the kernel [5]. The parameters of the kernel were optimized using the $fmin\_l\_bfgs\_b$ algorithm [22].

3.3.2 Acquisition Function. We use the Expected Improvement (EI) acquisition function. It calculates the expectation at a new point, $f(x)$, greater than the current best so far, $f(x^*)$:

$$\begin{aligned} EI(x) &= \mathbb{E}[max(0, (f(x) - f(x^*)))] \\ &= \begin{cases} (\mu(x) - f(x^*) - \zeta)\Phi(Z) + \sigma(x)\phi(Z), & \text{if } \sigma(x) > 0 \\ 0, & \text{if } \sigma(x) = 0 \end{cases} \end{aligned} \tag{8}$$

Where,

$$Z = \begin{cases} \frac{\mu(x) - f(x^*) - \zeta}{\sigma(x)}, & \text{if } \sigma(x) > 0 \\ 0, & \text{if } \sigma(x) = 0 \end{cases} \tag{9}$$

$\mu(x)$ and $\sigma(x)$ are the mean and standard deviation, respectively. Φ and ϕ are the CDF and PDF of the standard normal distribution, respectively [5]. ζ controls the amount of exploration during the optimization, whose higher value is responsible for more exploration. We use ζ=0.01 in our case.

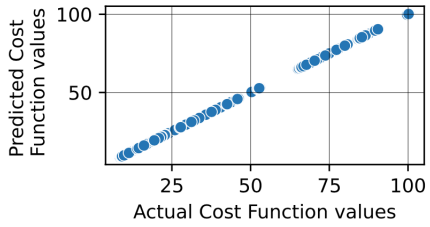

Figure 4: DecsisionTreeRegressor accuracy

Combined with the GPRegressor, which tests a range of candidate samples from the domian, the EI acquisition function scores each of the samples and keeps the scores that tend to maximize the cost function and is worth to be evaluated with the expensive cost function. We call these Bayes suggested points.

3.3.3 Machine Learning (ML) predictor. We need to evaluate the expensive cost function, which is the circuit simulation with CAD tools, such as HSPICE, PrimeTime, NanoTime, etc at Bayes suggested points. However, simulation using the CAD tool is highly time and resource constrained. To circumvent this we train an ML model to predict the cost function values at the Bayes suggested points. This ML predictor is used to evaluate the costly objective function at the Bayes suggested points with very high accuracy.

4 RESULTS AND ANALYSIS

In this section, we analyze our custom dataset and evaluate the performance of our ML predictor and the Bayes optimizer.

4.1 Dataset Analysis

Throughput increases as the number of AI chiplet increases (Fig. 3 (a)). However, fewer AI chiplets tend to maximize the cost function because of the area and latency increase (Fig. 3 (a,c,d)). Please note that other features, such as the number of CPU/HBM chiplets and

Table 4: Optimized design choices

AI-chiplet dim	HBM count	HBM location.	Package type	Data Rate/pin(gbps)	IO density/mm^2
2x2	3	left+right+ top/bottom (3D-stacked)	FOVEROS	20	110000

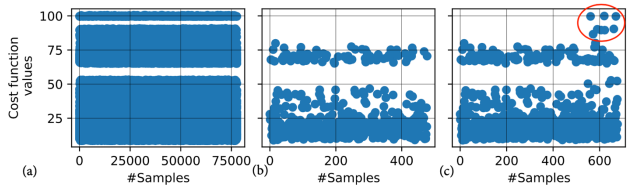

Figure 5: (a) Total state space (search space) with huge sample size, (b) Initial dataset to the optimizer with only 500 samples, (c) Updated dataset after 200 iterations of the optimizer. Optimizer finds the points (at the top right corner circled in red) that maximize the cost function.

their location, impact the final cost function. Because of the space constraint, we are not able to illustrate all of the contributing factors to the final cost function with the figure.

In the packaging space, the 3D packaging, FOVEROS and SoIC, have the higher bandwidth (3 (e)) due to their higher data rate and denser IO density compared to the 2.5D packagings. 3Ds also has lower energy compared to 2.5D (Fig. 3 (f)). However, they have higher cost & complexity compared to 2.5D (Fig. 3 (g)). Overall, the highest bandwidth and the lowest energy of FOVEROS outweigh its highest cost & complexity to maximize the final cost function (Fig. 3 (h)).

4.2 ML Predictor

We train a DecisionTreeRegressor on our custom dataset of sample size ~ 50k. On the test set of sample size ~ 40k, it achieves a mean absolute error of 0.0014 and a r^2 score of 0.99. Fig 4 shows the actual cost function values and the predicted cost function values on the test data almost overlap each other. We use *absolute error* as the loss function and expand the tree nodes until all leaves are pure to reach the desired accuracy. We aim for higher accuracy as we use this model to mimic the real simulation with CAD tools.

4.3 Optimizer

Fig. 5 (a) shows the state space we consider for this work. In reality, it can be even larger. Fig. 5 (b) shows the initial dataset with only 500 samples (a small subset of the total state space) to our optimizer. The initial dataset does not have the points that maximize the cost function. However, with only 200 iterations, the optimizer locates the points that maximize the cost function (Fig. 5 (c)). The run time of the optimizer is also very low, in *seconds* range. The set of points (i.e., design choices) that our optimizer suggests as the optimum so far are presented in Table 4.

5 CONCLUSION

In this study, we perform System and Design Technology Co-optimization (STCO & DTCO) of AI chiplet-based accelerator by exploring several domains of chiplet-based architecture to find the optimum design choices. We define a vast search space, create a custom dataset within the space, and locate the design choices that maximize our cost function with the help of the optimizer. Our optimizer demonstrates resource and time consumption efficiency by locating the global maxima within the search space with only 200 iterations and a run time of *sec* range.

6 ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Grant Number CRII-2153394.

REFERENCES

[1] 2015. *High-Bandwidth Memory(HBM).* Retrieved January 2, 2023 from https://www.amd.com/system/files/documents/high-bandwidth-memory-hbm.pdf
[2] 2021. *Heterogeneous Integration Roadmap.* Retrieved January 2, 2023 from https://eps.ieee.org/technology/heterogeneous-integration-roadmap/2021-edition.html
[3] Chen, F.C. et al. 2019. System on Integrated Chips (SoICTM) for 3D Heterogeneous Integration. In *2019 IEEE 69th Electronic Components and Technology Conference.*
[4] Coskun, Ayse et al. 2022. Cross-Layer Co-Optimization of Network Design and Chiplet Placement in 2.5-D Systems. In *2020 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, Vol. 39.
[5] Peter I. Frazier. 2018. A Tutorial on Bayesian Optimization. arXiv:arXiv:1807.02811
[6] Gomes, Wilfred et al. 2020. 8.1 Lakefield and Mobility Compute: A 3D Stacked 10nm and 22FFL Hybrid Processor System in 12×12mm2, 1mm Package-on-Package. In *2020 IEEE International Solid- State Circuits Conference.* 144–146.
[7] Kenny Cheng-Hsiang Hsieh. 2021. Designs of Communication Circuits for Side-by-Side and Stacked Chiplets. In *ISSCC 2021 Forums.*
[8] Hwang, Ranggi et al. 2020. Centaur: A Chiplet-based, Hybrid Sparse-Dense Accelerator for Personalized Recommendations. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture.* IEEE Computer Society, Los Alamitos, CA, USA, 968–981.
[9] Ingerly, D. B. et a;. 2019. Foveros: 3D Integration and the use of Face-to-Face Chip Stacking for Logic Devices. In *2019 IEEE International Electron Devices Meeting.*
[10] Hong Jiang. 2022. Intel's Ponte Vecchio GPU : Architecture, Systems Software. In *2022 IEEE Hot Chips 34 Symposium.* 1–29.
[11] Kao, Sheng-Chun et al. 2020. ConfuciuX: Autonomous Hardware Resource Assignment for DNN Accelerators using Reinforcement Learning. arXiv:2009.02010
[12] Krishnan, Srivatsan et al. 2022. Multi-Agent Reinforcement Learning for Microprocessor Design Space Exploration. arXiv:2211.16385
[13] Kumar, Aviral et al. 2022. DATA-DRIVEN OFFLINE OPTIMIZATION FOR ARCHITECTING HARDWARE ACCELERATORS. arXiv:2110.11346
[14] Lee, Frank J. C. et al. 2020. Heterogeneous System-Level Package Integration — Trends and Challenges. In *2020 IEEE Symposium on VLSI Technology.* 1–2.
[15] Lin, Mu-Shan et al. 2019. A 7nm 4GHz Arm®-core-based CoWoS® Chiplet Design for High Performance Computing. In *2019 Symposium on VLSI Circuits.* C28–C29.
[16] Lin, Ting-Ru et al. 2020. A Deep Reinforcement Learning Framework for Architectural Exploration: A Routerless NoC Case Study. In *2020 IEEE International Symposium on High Performance Computer Architecture.* 99–110.
[17] Mahajan, Ravi et al. 2016. Embedded Multi-die Interconnect Bridge (EMIB) – A High Density, High Bandwidth Packaging Interconnect. In *2016 IEEE 66th Electronic Components and Technology Conference.* 557–565.
[18] Kaniz Mishty and Mehdi Sadi. 2021. Designing Efficient and High-Performance AI Accelerators With Customized STT-MRAM. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 10, 1730–1742.
[19] Naffziger, Samuel et al. 2020. 2.2 AMD chiplet architecture for high-performance server and desktop products. In *2020 IEEE International Solid-State Circuits Conference-(ISSCC).* IEEE, 44–45.
[20] Naffziger, Samuel et al. 2021. Pioneering Chiplet Technology and Design for the AMD EPYC™ and Ryzen™ Processor Families : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture.* 57–70.
[21] John Park. 2022. 3D-IC Design Challenges and Requirements. In *Cadence.*
[22] Pedregosa, F. et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[23] Shao, Yakun Sophia et al. 2019. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture.* 14–27.
[24] Tan, Zhanhong et al. 2021. NN-Baton: DNN Workload Orchestration and Chiplet Granularity Exploration for Multichip Accelerators. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture.* 1013–1026.
[25] Wang, Tianqi et al. 2022. Application Defined On-chip Networks for Heterogeneous Chiplets: An Implementation Perspective. In *2022 IEEE International Symposium on High-Performance Computer Architecture.* 1198–1210.
[26] Wu, Yibo et al. 2022. Upward Packet Popup for Deadlock Freedom in Modular Chiplet-Based Systems. In *2022 IEEE International Symposium on High-Performance Computer Architecture.* 986–1000.
[27] Zhang, Dan et al. 2022. A Full-Stack Search Technique for Domain Optimized Deep Learning Accelerators. arXiv:2105.12842