Floorplet: Performance-Aware Floorplan Framework for Chiplet Integration

Shixin Chen, Shanyi Li, Zhen Zhuang, Su Zheng, Zheng Liang, *Graduate Student Member, IEEE*, Tsung-Yi Ho, Bei Yu, *Senior Member, IEEE*, and Alberto L. Sangiovanni-Vincentelli

***Abstract*—A chiplet is an integrated circuit (IC) that encompasses a well-defined subset of an overall system's functionality. In contrast to traditional monolithic system-on-chips (SoCs), chiplet-based architecture can reduce costs and increase reusability, representing a promising avenue for continuing Moore's law. Despite the advantages of multichiplet architectures, floorplan design in a chiplet-based architecture has received limited attention. Conflicts between cost and performance necessitate a tradeoff in chiplet floorplan design since additional latency introduced by advanced packaging can decrease performance. Consequently, balancing performance, cost, area, and reliability is of paramount importance. To address this challenge, we propose floorplan chiplet (Floorplet), a framework comprising simulation tools for performance reporting and comprehensive models for cost and reliability optimization. Our framework employs the open-source Gem5 simulator to establish the relationship between performance and floorplan for the first time, guiding the floorplan optimization of multichiplet architecture. The experimental results show that our method decreases interchiplet communication costs by 24.81%.**

***Index Terms*—2.5-D integrated circuits (ICs), chiplet floorplanning, computer architecture, hardware/software co-design.**

I. Introduction

HOW CAN we design complex electronic systems with high performance and low cost? This is a fundamental challenge that has driven the development of integrated circuit (IC) design for the past 50 years. Moore's law [1] has been the dominant paradigm that describes how advances in chip technology improve chip performance and reduce cost by integrating more transistors on a single die. However, as the cost of finer lithography patterns increases, the economic benefits of Moore's law are diminishing. Therefore, many foundries, such as TSMC, Samsung, and Intel, are exploring alternative solutions to reduce wafer costs and improve yields [2]. One of the promising solutions is the utilization of advanced heterogeneous integration and multichiplet architecture.

Manuscript received 3 August 2023; revised 2 November 2023; accepted 10 December 2023. Date of publication 28 December 2023; date of current version 21 May 2024. This work was supported in part by The Research Grants Council of Hong Kong, SAR, under Project CUHK14210723. This article was recommended by Associate Editor C. Zhuo. *(Corresponding author: Shixin Chen.)*

Shixin Chen, Shanyi Li, Zhen Zhuang, Su Zheng, Tsung-Yi Ho, and Bei Yu are with the Department of Computer and Science, The Chinese University of Hong Kong, Hong Kong (e-mail: sxchen22@cse.cuhk.edu.hk).

Zheng Liang and Alberto L. Sangiovanni-Vincentelli are with the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720 USA.

Digital Object Identifier 10.1109/TCAD.2023.3347302

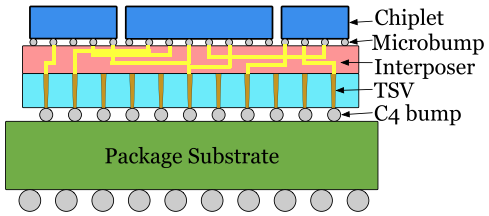


Fig. 1. Architecture of chiplet-based 2.5-D package.

A chiplet is an IC with a specific function obtained by partitioning a traditional monolithic system-on-chip (SoC). Chiplets can also be considered intellectual property (IP) components for reuse in multiple systems to reduce design costs. Additionally, chiplets provide an opportunity for heterogeneous integration with different technology nodes [3], [4], [5]. In chiplet-based architecture, less important components can use cheaper technology nodes. In contrast, some specified components like analog, power, and memory modules can use more advanced technology to reduce overall costs further and improve yield [6].

The chiplet-based design method needs advanced package techniques to utilize its characteristics fully. Chiplets can be assembled in three dimensions (i.e., 3-D ICs) or placed on a silicon interposer (i.e., 2.5-D ICs). Fig. 1 illustrates an example of advanced 2.5-D package using multichiplet architecture [7]. In the figure, the chiplets with various functions are placed on a silicon interposer using microbumps. Data communication between chiplets occurs through wires within the interposer. Then, the interposer is bonded to the package substrate with C4 bumps.

The multichiplet architecture has attracted significant attention from academia and industry due to the advantages mentioned above. However, challenges and drawbacks are also associated with chiplets that hinder their popularity in practical usage. If not handled properly, these challenges may result in decreased performance and higher fabrication costs instead of reducing costs.

First, from a performance perspective, the multichiplet architecture can suffer from degraded performance due to the extra physical wirelength between chiplets on the interposer. Interchip nets are routed on the redistribution layers (RDLs) by chip-scale wires [8]. Therefore, deciding the proper locations of chiplets on the interposer, i.e., floorplan design of multiple

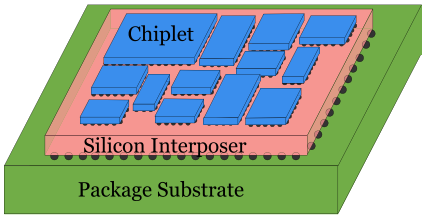


Fig. 2. Floorplan design of chiplets in 2.5-D package.

chiplets shown in Fig. 2, will greatly impact the communication between chiplets. Consequently, the overall performance of chiplet-based architecture is highly dependent on the quality of floorplan design.

Second, from a cost standpoint, the use of the advanced package can introduce more reliability issues. The reliability issues of the 2.5-D package will potentially affect IP functionality and reduce the service life of the chiplet system. Due to a higher degree of mismatch in the coefficient of thermal expansion (CTE) in advanced packages, the bump reliability may affected by stress, and chiplet cracking may be caused by package warpage [9], [10], [11].

Third, from design automation tools, design methodologies and electronic design automation (EDA) tools do not provide adequate support to practical advanced 2.5 packages specifically. Much literature on the chiplet-based floorplan design is based on abstract chiplets without specific functionality. In [6], the automated design of the chiplet was analyzed from a high-level architecture perspective, and the results have not been tested in realistic circuit designs. Prior art [12] focuses on partitioning SoC into chiplets based on cost analysis. Zhuang et al. [2] simply considered package reliability and ignored performance degradation caused by extra package cost and die-to-die interfaces. Therefore, designing practical methodologies and EDA tools based on actual circuits is important to boost the performance of chiplet.

To address these challenges and drawbacks, we propose Floorplet, a performance-aware framework for co-optimizing floorplan and performance of chiplet-based architecture. Unlike previous papers that use abstract chiplets without specific functionality or ignore performance metrics in floorplan design, we consider realistic chiplets with complex data flow and incorporate performance factors into floorplan optimization. Our framework consists of three main components with corresponding contributions as follows.

1) *parChiplet:* An algorithm to partition realistic SoC into functional chiplets that can be fabricated and analyzed by EDA tools. The chiplets have specific functions so that we can bring more hardware information to enable an analysis of complex data flow in chiplet-based architecture.
2) *simChiplet:* A simulation platform based on Gem5 simulator [13] that evaluates the performance impact of different floorplan solutions for chiplet-based architecture. The communication latency of chiplet-to-chiplet depends on the latency in the interposer, which builds a connection between the floorplan design and architecture performance.
3) *optChiplet:* A floorplan framework for chiplet architecture that considers reliability, cost, and area in addition to performance metrics. We tested our work on various benchmarks to show that it outperforms previous methods and achieves co-optimization of architecture and technology.

The remainder of this article is organized as follows. Section II introduces the cost and reliability model for the chiplet-based architecture and important issues in floorplan design. Section III provides detailed components of Floorplet and corresponding algorithms. Section IV conducts experiments to demonstrate the superiority of Floorplet and analyzes experimental results. Finally, Section VI concludes this article.

II. Preliminaries

A. Chiplet and Cost Model

Chiplet Definition: A chiplet is a small chip that is obtained by partitioning a large monolithic SoC chip. Chiplets can be fabricated using different technologies and integrated into a larger system using advanced package techniques like interposer-based 2.5-D packages. Chiplets can offer several benefits over monolithic SoC chips, such as lower fabrication and design costs, higher yield, and better performance. At the same time, the chiplet-based architecture also poses several challenges and drawbacks that need to be addressed. One of the main challenges is how to optimize the floorplan of chiplets on the interposer, which affects the performance, reliability, cost, and area of the system.

Yield and Defect Density: According to Moore's law, the number of transistors in an IC doubles every 18–24 months, which has been well proved in the past half-century [1]. However, the size of transistors in IC has now met its physical limit and cannot be reduced as in the past [14]. Therefore, advanced package techniques like interposer-based 2.5-D packages are proposed to address this issue to build large-scale systems [15]. Compared to a large monolithic SoC chip, the 2.5-D package contains many smaller chiplets, which are partitioned from the monolithic SoC. With chiplets, a larger system can be constructed, and the fabrication and design costs will be reduced substantially.

One of the main factors that affect the fabrication cost is yield, which is the probability that a chip die is functional and free of defects. The yield depends on the die area and the defect density of the technology. A widely used model to estimate yield is the negative binomial model [16]

$$Y(s) = \left(1 + \frac{d_0 s}{\alpha}\right)^{-\alpha} \tag{1}$$

where Y is the yield, s is the die area, d_0 is the defect density, and α is the cluster parameter. In the 7-nm technology, the typical value of d_0 is 0.09 cm^{-2}, and the typical value of α is 10 [15]. With this equation, the manufacturing costs can be estimated based on the processed wafer's yield Y and unit price P_0. Fig. 3 shows that the yield decreases and the cost increases as the area increases, especially for advanced technologies with higher defect density.

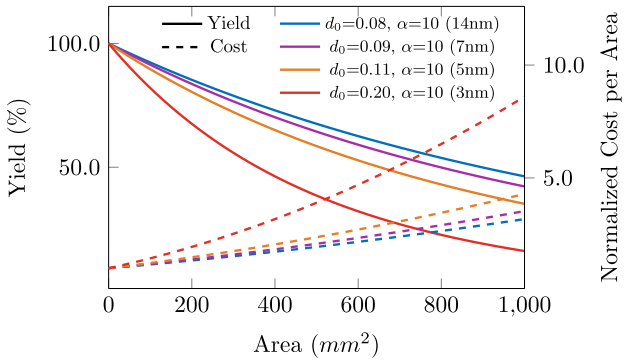


Fig. 3. Relation between yield/cost and area of different technologies. The figure shows that the yield decreases and the cost increases as the area increases, especially for advanced technologies with higher defect density.

The Taylor expansion can be used to approximate the manufacturing cost per yielded area as follows:

$$y(s) = \frac{P_0}{Y} \approx P_0\left(1 + d_0 s + \frac{\alpha - 1}{2\alpha} d_0^2 s^2\right) \tag{2}$$

where $y(s)$ is the manufacturing cost per yielded area, which will rise quickly as the die sizes increase. This equation implies that with smaller chiplets partitioned from a monolithic SoC, $y(s)$ can be lower, thereby decreasing overall manufacturing expenses.

Cost Model of 2.5-D Package: In addition to the manufacturing cost of chiplets, the bonding cost and yield of chiplets and the interposer in the 2.5-D package should be considered as well. As shown in Fig. 2, the raw cost of a 2.5-D package consists of several components: raw chiplets C_{chip}, raw substrate C_{sub}, raw package C_{pack}, raw silicon interposer C_{inter}, and bonding cost C_{bond} of each chiplet. The bonding process involves attaching chiplets to an interposer and the interposer to a substrate using micro-bumps or through-silicon vias (TSVs). The bonding process may introduce defects or failures that affect the functionality of the package.

Therefore, by considering raw cost and bonding cost, the overall cost of the 2.5-D package can be expressed as follows:

$$C_{\text{package}} = C_{\text{pack}} + C_{\text{inter}} \times \left(\frac{1}{y_1 \times y_2^n \times y_3} - 1\right) + C_{\text{sub}} \times \left(\frac{1}{y_3} - 1\right) \tag{3}$$

where n is the number of chiplets, y_1 is the yield of the interposer, y_2 is the bonding yield of chiplets, and y_3 is the bonding yield of interposer.

The overall cost of the 2.5-D package is

$$C_{2.5D} = \frac{\frac{C_{\text{pack}}}{Y_{\text{pack}}} + \sum_{i \in n}\left(\frac{C_{\text{chip\_}i}}{Y_{\text{chip\_}i}} + C_{\text{bond\_}i}\right)}{Y_{\text{bond}}^n} \tag{4}$$

where Y_{pack}, Y_{chip}, and Y_{bond} are, respectively, the yield of the package, the yield of chiplets, and the yield of the bond. The values of each term in our cost model are validated based on data from public information and in-house sources [15].

B. Reliability Issues of 2.5-D Package

Warpage: Warpage is a major reliability concern in advanced packages that refers to abnormal bending of shape due to the mismatch in the CTE of different materials. Warpage can cause the package to bend and result in the cracking of chiplets and substrates. The critical parameters affecting warpage include mold thickness, molding materials, and the ratio of chiplet to package area [9], [10], [11]. Warpage can be measured by the variation in vertical height at different positions on a package. In this work, the warpage from the center to the edges of packages can be controlled by an effective warpage computing model introduced in previous works [17], [18]. The model is given as follows:

$$w(x) = \frac{t \cdot \Delta\alpha \cdot \Delta T}{2 \cdot \lambda \cdot D}\left[\frac{1}{2}x^2 - \frac{\cosh(kx) - 1}{k^2 \cosh(kl)}\right] \tag{5}$$

where $\Delta\alpha$ represents the difference in CTE between the chiplets and the substrate, and ΔT represents the thermal load. The coefficients t, λ, D, and k are related to the material properties. The center of the package is used as the origin for building this model. The variables x and l represent half the length of a chiplet and half the length of a substrate, respectively.

Bump Stress: Bump stress is another major reliability concern in advanced packages that refers to the mechanical stress induced by bumps during the bonding process. Bump stress can increase the risk of bump cracks or component deformation, which can affect the functionality of the package. Bump stress depends on the location and size of bumps, as well as the placement and shape of components near the bumps.

Hotspot bumps are defined as the bumps at the edge of the silicon interposer that have a higher risk of deformation than the bumps in the interposer center with uniform stress. Analysis has shown that the edges of bumps experience high stress due to the proximity of components, especially for hotspot bumps that have a higher possibility of deformation or crack [19], [20]. In this work, the bump stress can be controlled by avoiding the overlap between chiplets and hotspot bumps. Specifically, margin space around hotspot bumps is ensured to prevent chiplets from overlapping with each other and reduce bump stress.

C. Floorplan of Chiplets

The floorplan is an important stage in designing chiplet-based systems that allocates the major functional blocks on the interposer. The floorplan affects the overall area, the routing wirelength, the thermal effects, and the reliability of the design. Prior arts [21], [22] consider the area and wirelength of the floorplan, while ignoring the performance metrics, such as the data movement frequency and communication latency between different chiplets. If the optimization of the floorplan can consider the performance metrics, the final performance of the chiplet system can be improved.

Li et al. [23] utilized the modular design scheme, which means that each chiplet is placed on the tiles with an equal size on the interposer. The scheme improves the reusability

of each chiplet IP, while it may increase the potential of extensive interposer area cost due to some tiles being free on the interposer. In our framework, a different scheme of chiplet floorplanning is utilized. Routers in the interposer are not involved in the floorplan scheme in our work, instead, more flexibility is added to the routing wire in the interposer and minimize the interposer area. In this way, the chiplet-based architecture can benefit from the router-free scheme with a larger floorplan solution space to obtain the optimal floorplan solution.

The chiplet-based architecture makes it possible to reduce time to market by utilizing existing IPs. By combining various chiplets with different functions in the 2.5-D package, design companies can create new electronic systems without starting from designing and testing to manufacturing SoC. For example, in the recent AMD ZEN 2 architecture [24], server and desktop processors can share the identical chiplet named core complex die (CCD), which contains the CPU cores and caches. Given such existing chiplets, it is challenging to achieve a tradeoff between cost and performance. In our framework, we combine the performance metric with the floorplan of the chiplet by building a chiplet simulation platform to evaluate the chiplet design, which will be elaborated in Section III.

D. Problem Formulation

The input of our framework is a set of chiplets $\mathcal{C}$ partitioned from a monolithic SoC, a netlist N that connects the chiplets, and the bump positions of the silicon interposer. The output the framework is a floorplanning solution that provides the location and orientation of each chiplet on the silicon interposer. The objective of the optimization is

$$\begin{aligned} &\min \quad \beta_1 \times wl + \beta_2 \times \alpha_p + \beta_3 \times C_{2.5D} \\ &\text{s.t.} \quad wpg_p \leq wpgt_p \end{aligned} \tag{6}$$

where wl represents half-perimeter wirelength (HPWL) between chiplets, α_p represents the warpage of the package, and $C_{2.5D}$ is the cost of 2.5-D package from (4). β_i represents a user-defined coefficient that can be modified for a tradeoff between multiple objectives.

During optimization, three types of constraints are considered.

1) *Overlap Constraints:* Each chiplet cannot overlap with other chiplets or hotspot bumps.
2) *Warpage Constraints:* As shown in (6), the warpage of the package p cannot exceed a threshold in both the x-axis and y-axis directions.
3) *Bump Margin Constraints:* A margin space is defined around hotspot bumps to avoid overlap with chiplets and reduce bump stress.

III. Floorplet Framework

A. Overview of the Framework

Fig. 4 shows the overview of the proposed Floorplet, which consists of three main steps: 1) chiplet partitioning; 2) floorplan optimization; and 3) performance evaluation.

First, we propose the *parChiplet* algorithm to partition a monolithic SoC into a set of chiplets with different functions and area constraints. The algorithm recursively divides the hierarchical tree structure that represents the SoC components and their connections.

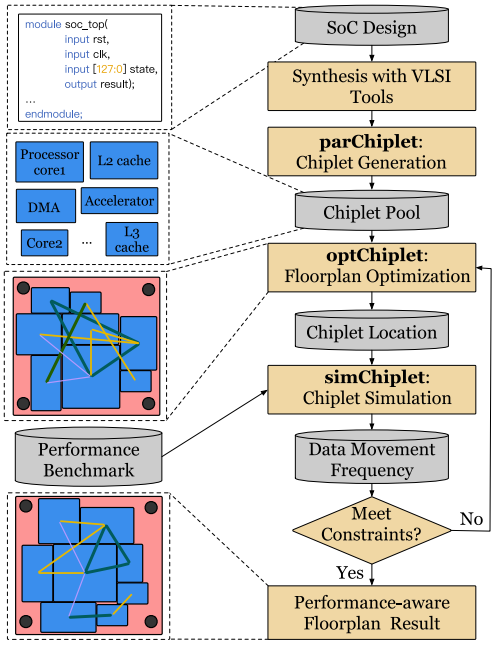


Fig. 4. Overall flow of the Floorplet.

Second, we propose *optChiplet* to formulate the floorplan of the chiplet as a mathematical programming (MP) problem that considers multiple objectives, such as package cost, wirelength, warpage, and bump stress. We solve the MP problem to obtain a primary floorplan solution that provides the location and orientation of each chiplet on the silicon interposer.

Third, we evaluate the performance of the chiplet design using *simChiplet* built based on Gem5 garnet3.0 [13]. The simulation platform can report the data movement frequency and latency of different chiplet pairs based on various benchmarks. The data movement frequency is utilized as feedback to further optimize the floorplan results. In the simulation platform shown in Fig. 5, the data are sent or received in packages, which have an equal volume. In this way, more data movement frequency can be seen as more data movement volume. The details of each step will be elaborated as follows.

B. parChiplet: Chiplet Generation Method

Some previous works [2], [6] target highly abstract chiplets without specific functions, while we choose an actual hardware design of SoC and utilize synthesis tools in very-large-scale integration (VLSI) flow to obtain the area and netlist of the design. Then, based on various functions and area information, an algorithm is developed to partition chiplets from the SoC and obtain a chiplet pool.

Zhuang et al. [2] do not discuss the integrity of chiplet functions when obtaining chiplets, i.e., some circuit macros that communicate tightly may be partitioned to distinctive chiplets. To bring more practical design information into chiplet design, we use Python language to design a script to process the

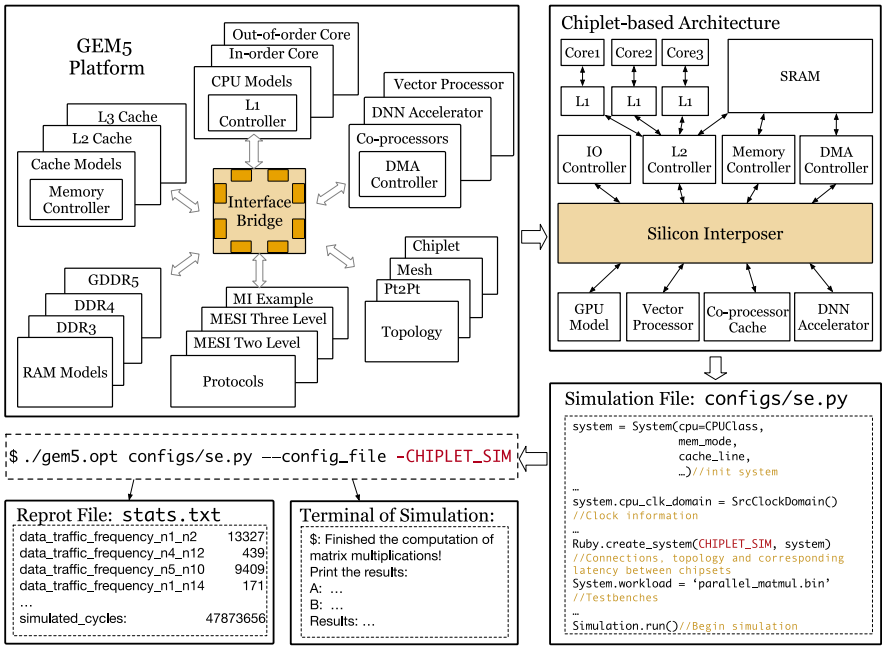


Fig. 5. Simulation flow embedded into Gem5 platform.

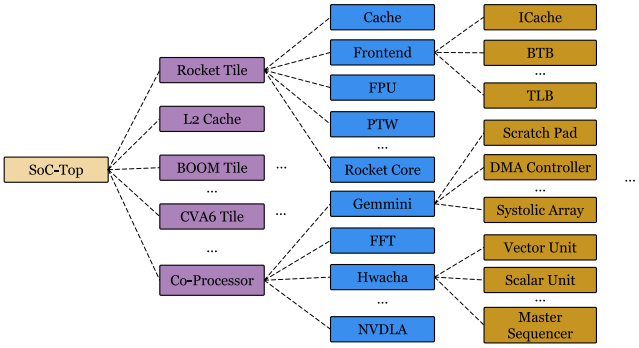


Fig. 6. Hierarchical tree of SoC components.

synthesis result of SoC from VLSI tool like Hammer [25]. The output of the script is a hierarchical tree of SoC, and an example is shown in Fig. 6. The hierarchical tree takes functional integrity and area into consideration. In this way, the partitioned chiplets have relatively independent functions that can be reused as IPs by other designs. Meanwhile, the number of chiplets divided from the SoC system can be controlled within an acceptable range.

Furthermore, the granularity of the partition is very important during constructing a chiplet pool. On the one hand, if the chiplet area is too large like a part of a monolithic SoC, the benefit from the decreasing of area cost will be eliminated. On the other hand, if the chiplet area is too small like a fine-grained fragmentation with incomplete function, it is impossible for fabrication to provide such an IP.

Therefore, the rule is given to make sure the area range$(a_{\min}, a_{\max})$ is given as

$$a_{\min} = \frac{a_{\text{SoC}} \times rr}{|\mathcal{C}|_{\max}}, \quad a_{\max} = \frac{a_{\text{SoC}} \times rr}{|\mathcal{C}|_{\min}} \tag{7}$$

Algorithm 1 parChiplet$(T, a_{\min}, a_{\max})$

Input: $(T, a_{\min}, a_{\max})$, T is the hierarchical tree structure like Fig. 6, $a_{\min}$ is the minimal threshold of chiplet area, $a_{\max}$ is the maximal threshold of chiplet area.

Output: The chiplet pool $\mathcal{C}$.

$\mathcal{C} = \emptyset$;
$N =$child(T); ▷ child$(\cdot)$ gives the children nodes of T.
$n_r = \emptyset$
for $n_i \in N$ **do**
  if area$(n_i) > a_{\max}$ **then** ▷ area$(\cdot)$ gives node's area.
    parChiplet$(n_i, a_{\min}, a_{\max})$;
  else if area$(n_i) < a_{\min}$ **then**
    $n_r =$comb(n_i, n_r); ▷ comb$(\cdot)$ combines two nodes.
  else
    $\mathcal{C} = \mathcal{C} \bigcup \{n_i\}$;
  end if
$\mathcal{C} = \mathcal{C} \bigcup \{n_r\}$;
end for
return $\mathcal{C}$;

where $|\mathcal{C}|_{\min}$ and $\mathcal{C}|_{\max}$ represent the typical minimal and maximal number of chiplets in the chiplet-based architecture, respectively. a_{SoC} and rr represent the overall area of the monolithic SoC given by the synthesis tool and the relaxation ratio of a_{SoC}, respectively. rr is set to make sure to avoid the failure of the chiplet partition. The total area of partitioned chiplets should be larger than the original a_{SoC} to avoid failure of the floorplan because extra areas of interfaces die-to-die are introduced.

By combining all analyses into an algorithm, we propose Algorithm 1. The algorithm is straightforward to process the SoC hierarchical tree recurrently. The input of the algorithm is the synthesis results given by Hammer [25] from different SoC designs. For various SoC designs, once given the synthesis results, the hierarchical tree can be constructed automatically with function integrity and area information. Since we have

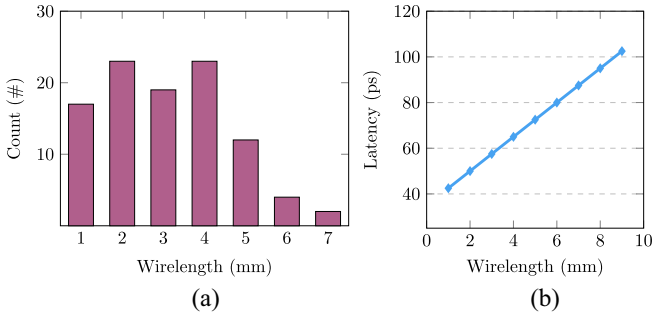


Fig. 7. Latency-wirelength model. (a) Distribution of the wirelength between chiplets. (b) Latency weight versus wirelength relation.

designed the parser to analyze the chiplet interconnection relationship, the algorithm can be generalized across different SoC.

C. simChiplet: Chiplet Simulation Method

In this step, we build a platform based on Gem5 garnet3.0 [13] to evaluate the data movement frequency of the chiplet-based architecture and the overall latency of running various workloads. The detailed simulation flow is shown in Fig. 5, where we set the characteristics of various hardware components to mimic the functionality of the original monolithic SoC. In the original simulation tool, network-on-chip (NoC) in Gem5 garnet3.0 combines different modules in the chiplet-based architecture. The NoC module in gem5 is used to record the data request and data volume, which can assist in obtaining the data movement between chiplets. However, the original platform does not support setting various values of latency between different chiplet-pair flexibly. Instead, we improve the flexibility of the original topology by embedding the latency weight given by the latency-wirelength model in Fig. 7(b), which is going to be introduced as follows.

In the 2.5-D package, the interchiplet communication latency is determined by the length of the routed wires inside the interposer, the microbumps, and the die-to-die interfaces. In the monolithic SoC, the communication latency is determined by the critical path in the circuit with a typical max wirelength of about 1.4 mm, while in the chiplet-based architecture, the max wirelength can be much longer than that of SoC. For example, in the chiplet-based architecture containing 64 cores [26], the max wirelength can reach about 10 mm.

The latency introduced by the chiplet architecture cannot be ignored, as it can affect the performance of the system. In UCIe 1.0 [27], an open industry standard for on-package connectivity between chiplets, the latency of the interface should be smaller than 2 ns. If the system runs with 2 GHz, extra clock cycles will be introduced to chiplet communication. To show the latency influence, we choose an SoC containing two Rockets cores and two BOOM cores and use their HPWL distribution to estimate their latency weights as shown in Fig. 7.

Kim et al. [26] designed and constructed a chiplet-based 64-core processor to illustrate the chiplet design flow. It built and verified the interposer delay model consisting of resistance, inductance, conductance, and capacitance. According to the results, in the 0.2–10.0 mm length range, as the wire becomes longer, both communication delay and energy increase linearly. In our experiment, the wire length falls into the wire length range mentioned above, so we assume that it is reasonable to utilize the wirelength-latency model in our framework. Therefore, we suggest using this conclusion to build our interposer delay model.

Therefore, according to the HPWL between the chiplets on the silicon interposer and the relationship between the latency and wirelength [26], the wirelength is mapped into six ranges in Fig. 7(a), and each range has a specific latency weight. With the latency-wirelength model in Fig. 7(b), the file containing latency information and connections of chiplet is embedded into our simulation platform with an option `-CHIPLET_SIM` in the command line shown in Fig. 5.

D. optChiplet: Floorplan Optimization Method

The goal of our floorplan framework is to optimize the placement of chiplets on the silicon interposer while minimizing wirelength and improving reliability. The simulation platform needs more time than the iteration time of the MP-solver to obtain the data movement frequency among chiplets. Therefore, combining the time-consuming simulation with the existing solver together is unpractical. To decrease the running time of the performance-aware floorplan solution, the optimization is divided into two stages, i.e., primary floorplan and performance-aware floorplan.

In this section, our MP models for solving the floorplan problem are presented. First, the primary floorplan model that considers the chiplet dimensions, locations, rotations, warpage, and bump stress is introduced. Then, we describe the performance-aware floorplan model that incorporates the data movement frequency between chiplets obtained from our simulation platform.

Primary Floorplan: The input data consists of n chiplets $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$, each with a fixed outline and a variable location and orientation. The center area of the silicon interposer is a bounding box $W \times H$, where we aim to place the chiplets without overlap. x_{c_k} and y_{c_k} are used to denote the x-coordinate and y-coordinate of the lower-left corner of chiplet c_k, respectively. w_{c_k} and h_{c_k} are used to denote the width and height of chiplet c_k, which depend on whether it is rotated or not.

To formulate the floorplan problem as an MP model, we introduce some auxiliary variables and constraints as follows. The notations used in this part are listed in Table I.

1) In practical chiplet-based architecture, there exists a small distance to allow routing wire between two near chiplets. After the floorplan stages, there are follow-up stages like routing stages for the final chiplet-based design fabrication. Therefore, we simplified the optimization process by omitting the small distance between two nearby chiplet. To prevent overlap between chiplets, the binary variables $p_{i,j}$ and $q_{i,j}$ are used to indicate the relative positions of chiplet

TABLE I
NOTATIONS USED IN OPTCHIPLET

Notations	Meaning
$\mathcal{C}$	all chiplets of the architecture
$c_k \in \mathcal{C}$	a chiplet k in the architecture
p	the package of the design
W, H	width and height of floorplan region
(x_{c_k}, y_{c_k})	coordinate of the lower-left corner of chiplet c_k
(x_{b_s}, y_{b_s})	coordinate of the center of bump b_s
w_{c_k}, h_{c_k}	width and height of chiplet c_k
$(p_{i,j}, q_{i,j})$	indication of relative position of c_i and c_j
r_k	indication of rotation of c_k
d_{cc}	radius of the circumscribed circle of chiplet
d_{mr}	radius of the margin region around each bump
wpg_p^x, wpg_p^y	warpage of package p in x-axis and y-axis

c_i and chiplet c_j. The nonoverlap constraints can be expressed as

$$x_{c_i} + w_{c_i} \le x_{c_j} + W \cdot \left(p_{i,j} + q_{i,j}\right) \tag{8}$$
$$y_{c_i} + h_{c_i} \le y_{c_j} + H \cdot \left(1 + p_{i,j} - q_{i,j}\right) \tag{9}$$
$$x_{c_i} - w_{c_i} \ge x_{c_j} - W \cdot \left(1 - p_{i,j} + q_{i,j}\right) \tag{10}$$
$$y_{c_i} - h_{c_i} \ge y_{c_j} - H \cdot \left(2 - p_{i,j} - q_{i,j}\right) \tag{11}$$
$$p_{i,j}, q_{i,j} \in \{0, 1\}, 1 \le i < j \le |\mathcal{C}|. \tag{12}$$

If $(p_{i,j}, q_{i,j}) = (0, 0)$, the chiplet c_i is constrained to place on the left of chiplet c_j; if $(p_{i,j}, q_{i,j}) = (0, 1)$, the chiplet c_i is constrained to place on the bottom of chiplet c_j; if $(p_{i,j}, q_{i,j}) = (1, 0)$, the chiplet c_i is constrained to place on the bottom of chiplet c_j; if $(p_{i,j}, q_{i,j}) = (1, 1)$, the chiplet c_i is constrained to place on the top of chiplet c_j. In other words, by optimizing this pair, the location of various chiplet can be changed to avoid overlap between chiplets.

2) To allow rotation of chiplets, another binary variables r_k is used to indicate whether chiplet c_k is rotated by 90 degrees or not. The width and height of chiplet c_k can be calculated as

$$w_k = r_k \cdot h_k^o + (1 - r_k) \cdot w_k^o \tag{13}$$
$$h_k = r_k \cdot w_k^o + (1 - r_k) \cdot h_k^o \tag{14}$$

where w_k^o and h_k^o are the original width and height of chiplet c_k, respectively.

To account for reliability issues caused by warpage and bump stress, continuous variables wpg_p^x and wpg_p^y are used to represent the warpage on the x-axis and y-axis directions of the whole package p, respectively. The bump constraints are shown in the lower-left corner of Fig. 9, where continuous variables d_{cc} and d_{mr} are used to represent the radius of the circumscribed circle of each chiplet and the radius of the margin region around each hotspot bump, respectively. The warpage constraints can be expressed as follows:

$$w(x) = \frac{t \cdot \Delta\alpha \cdot \Delta T}{2 \cdot \lambda \cdot D}\left[\frac{1}{2}x^2 - \frac{\left((kx)^2 + 1\right)/2 - 1}{k^2 \cosh(kl)}\right] \tag{15}$$

where t, $\Delta\alpha$, ΔT, λ, and D are physical parameters related to the packing materials and dimensions. The warpage in each direction can be calculated by plugging in the corresponding values of x. The warpage upper bound can be enforced as

$$wpg_p^x \le wpgt_p, \quad wpg_p^y \le wpgt_p \tag{16}$$

where $wpgt_p$ is a user-defined threshold for an acceptable warpage. The bump margin constraints can be expressed as follows:

$$\left(x_{c_i} - x_{b_s}\right)^2 + \left(y_{c_i} - y_{b_s}\right)^2 \ge (d_{cc} + d_{mr})^2 \tag{17}$$

where (x_{b_s}, y_{b_s}) are the coordinates of the center of hotspot bump b_s, and (x_{c_i}, y_{c_i}) are the coordinates of the center of chiplet c_i. This constraint ensures that there is enough spacing between each chiplet and each hotspot bump to avoid excessive stress.

The objective function of the primary floorplan model is to minimize a weighted sum of wirelength (wl), area (α_p), warpage (wpg_p^x and wpg_p^y), and cost of 2.5-D package ($C_{2.5D}$) with bump constraints, which can be expressed as

$$\beta_1 wl + \beta_2 \alpha_p + \beta_3\left(wpg_p^x + wpg_p^y\right) + \beta_4 C_{2.5D} \tag{18}$$

where β_1, β_2, β_3, and β_4 are user-defined coefficients that reflect different design priorities.

We solve this MP model using an off-the-shelf solver to obtain an initial floorplan solution that satisfies all the constraints and optimizes all the objectives. Fig. 8(a) shows a set of chiplets partitioned from a monolithic SoC with their connections represented by lines with different widths indicating their data movement frequency. Fig. 8(b) shows an example of a primary floorplan solution with eight chiplets placed on a silicon interposer.

Performance-Aware Floorplan: After obtaining the primary floorplan solution, we feed it into our simulation platform to evaluate its performance in terms of data movement frequency among chiplets, as shown in Fig. 9. The simulation platform models the application workload, communication patterns, and memory hierarchy of the chiplet-based architecture. The simulation will report a set of frequency values for each pair of chiplets, denoted by $F = \{f_1, f_2, \ldots, f_M\}$, $M = C_2^{|\mathcal{C}|}$ represents two-combination of $\mathcal{C}$.

These frequency values can be used as inputs for our performance-aware floorplan model, which aims to further optimize the placement of chiplets by reducing the latency between frequently communicating pairs. The performance-aware floorplan model has the same variables and constraints

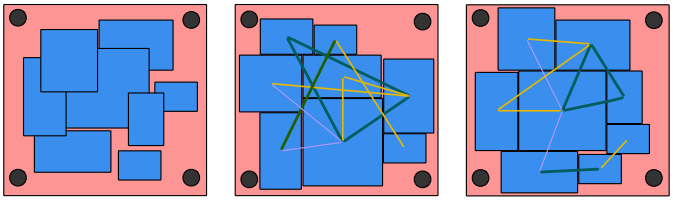


Fig. 8. Different floorplan designs of chiplet-based architecture. (a) Macros, bumps, and nets are generated from input data. (b) Floorplan design with eight chiplets without performance metrics. The wider line represents higher data movement frequency between chiplets. (c) Performance-aware floorplan design.

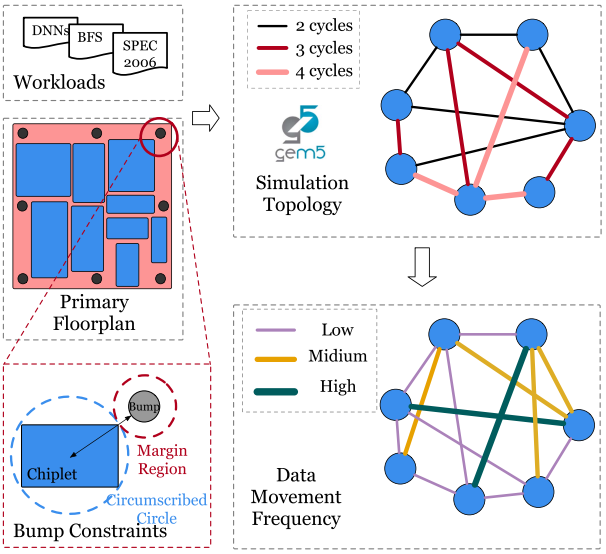


Fig. 9. Simulation framework for chiplet-based architecture. The blue circles represent chiplets, while the lines between them represent latency. After the simulation, the data movement frequency between chiplets will be reported, where the wider line represents a higher data movement frequency.

as the primary floorplan model, except for an additional term that reflects the data movement frequency

$$\beta_1 wl + \beta_2 \alpha_p + \beta_3 \left(wpg_p^x + wpg_p^y \right) + \beta_4 C_{2.5D} + \gamma_1 \sum_i f_i \quad (19)$$

where γ_1 is a user-defined coefficient that controls the tradeoff between data movement frequency and other objectives.

This MP model is also solved using an existing solver to obtain a final floorplan solution that balances multiple objectives and meets the performance constraints. Fig. 8(c) shows an example of a performance-aware floorplan solution with eight chiplets placed on a silicon interposer.

IV. Experiments and Analysis

In this section, we evaluate the effectiveness of our proposed floorplan framework on realistic chiplet-based architectures. We construct the framework with actual chiplets considering performance or reliability issues instead of using abstract rectangles to represent chiplets. We also analyze the tradeoff between different objectives and do some ablation studies to check the effectiveness of reliability constraints.

A. Benchmarks and Baseline

We take advantage of Chipyard [28], an open-source SoC generator tool, to generate various SoCs consisting of different hardware components, such as CPU cores (e.g., Rocket [29] and BOOM [30]), co-processors (e.g., Gemmini [31] and Hwacha [32]), etc. To obtain the area information of each chiplet, we utilize Hammer [25] tools with 7-nm standard cell library ASAP7 [33] to synthesize various SoC designs. The area information of each chiplet provides the input of *parChiplet* to guide the floorplan optimization.

Each SoC can be partitioned into about 10–30 chiplets based on their functions and build a chiplet pool consisting of about 300 chiplets. The chiplet pool is constructed with various SoCs to avoid time-consuming synthesis design. Once meeting the same chiplet components, i.e., IPs, in the new chiplet-based architecture, the physical information can be reused again. Building a chiplet pool provides an opportunity to reuse the synthesis results from Hammer to avoid the extra process of synthesizing.

TABLE II
Selected Chiplets From the Chiplet Pool Constructed From Various SoC Designs

Modules	Chiplets	Area (μm^2)
Cores	SmallRocket	561513
	MediumRocket	635068
	LargeRocket	878987
	SmallBOOM	1067891
	MediumBOOM	2660359
	LargeBOOM	4083778
Co-processors	Systolic Array	1330437
	Gemmini Accelerator_1	2162646
	Gemmini Accelerator_2	4354390
	Hwacha Accelerator	1335859
	FFT Accelerator	26445
Caches	DMA Controller	212861
	L2 Cache Bank_1	417095
	L2 Cache Bank_2	557209
	L3 Cache Bank_1	835814
	L3 Cache Bank_2	3608173

For practical chiplet fabrication, it is very time-consuming to decide on the specific design for each chiplet. Only after obtaining the final design of the chiplet, the width, and height can be fixed. For our experiment, we cannot carefully design each chiplet one by one in a limited time. Therefore, we simplified the process by generating chiplets with a random width/height ratio within a rational range. Our framework is capable of handling practical chiplet whatever the width/height ratio. Some selected chiplets are listed in Table II.

To boost the computation capability of the SoC architecture, different SoCs are designed specifically with the target of optimizing different applications. For instance, [34] specifically designs the architecture for optimizing deep neural network (DNN) applications on flagship mobile devices. [35] designs an SoC architecture for mapping medical applications. Therefore, to optimize the performance of these function-specific chiplet-based architectures, the corresponding workloads are chosen to do the evaluation.

For chiplet-based DNN accelerators, ResNet-15 [36] and MobileNet-v2 [37] are utilized as the workloads. For chiplet-based general-purpose processors, some representative workloads are utilized from the commonly used CPU benchmark SPEC2006 [38]. The benchmarks encompass a diverse set of applications with varying performance characteristics, effectively covering the states of all chiplets. The workloads for various chiplet-based architectures are listed in Table III.

We compare our framework with the MP-based solver method [2], which can give the primary floorplan solution without performance consideration. Some ablation studies are conducted to evaluate the effectiveness of warpage constraints and bump stress constraints. Our framework and the baseline

TABLE III
DIFFERENT CHIPLET-BASED ARCHITECTURE WORKLOADS FROM RISC-V BENCHMARK, SPEC2006, AND DNNS ORDERLY

Architectures	Benchmarks	Applications
$\mathcal{C}_8$	whetstone	computer benchmarks [39]
	fir2sim	DSP-oriented algorithms
	iir	DSP-oriented algorithms
	mt-vvad	ISA basic instructions
	add-int	ISA basic instructions
$\mathcal{C}_{16}, \mathcal{C}_{22}$	400.perlbench	email tools in Perl
	401.bzip2	file compression
	403.gcc	C language compiler
	445.gobmk	go game
$\mathcal{C}_{30}$	ResNet15	DNN workload [36]
	MobileNet-v2	DNN workload [37]
	Encoder Module	DNN modules [40]

methods are implemented in C++ and use Gurobi [41] as the MP solver. All experiments are conducted on a Linux machine with an Intel Xeon CPU (E5-2630 v2@2.60 GHz) and 256-GB RAM.

B. Experiment Setting and Results

The framework is tested on four SoC architectures with different configurations, which are partitioned into 8, 16, 22, and 30 chiplets represented by $\mathcal{C}_8, \mathcal{C}_{16}$, $\mathcal{C}_{22}$, and $\mathcal{C}_{30}$. The $\mathcal{C}_8, \mathcal{C}_{16}, \mathcal{C}_{22}$, and $\mathcal{C}_{30}$ are partitioned from a SmallRocket core with the corresponding caches and peripheral modules, a MediumBOOM core with the corresponding caches and peripheral modules, a LargeBOOM core with peripheral modules, and a Gemmini accelerator with the processor core, caches and corresponding DMA controllers.

The data movement frequency between chiplets is the average of various benchmarks that are suitable for the specific SoC design. The user-defined coefficients are set based on some pre-experiments to obtain a good tradeoff between multiple objectives and ensure convergence of optimization. According to pre-experiments, β_1, β_2, β_3, β_4, and γ_1 in (6) are set to 1, 10, 100, 1, and 1, respectively. The floorplan results demonstrate this setting can achieve a balance between multiple objectives.

In Table IV, the experimental results are listed, where HPWL, PA, WPG, ComCost, and AvgLat represent HPWL of chiplet routing, package area, warpage of package and interchiplet communication cost, and the average clock cycles of finishing the workloads, respectively. The communication cost is the multiplication of the data movement frequency with the wirelength between two chiplets. Longer wirelength will bring more latency in RDLs in chiplet-based architecture. If frequent data movement occurs between two chiplet with too long distances, the overall performance of the design will deduct heavily because data movement costs can bring more waiting time between components. Therefore, this metric can represent the communication cost of the chiplet-based architecture.

The floorplan framework can reduce communication costs by placing chiplets with high data movement frequency close to each other. As a result, the performance-aware floorplan can decrease the average clock cycles of finishing workloads by 13.18%. The column Latency in Table IV shows that the performance improvement is more significant for architectures with more chiplets.

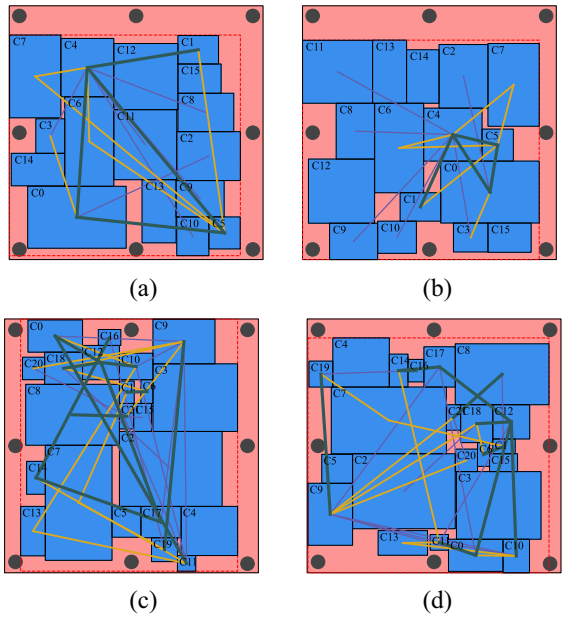


Fig. 10. Floorplan design of the chiplet-based architecture ($\mathcal{C}$=16) and ($\mathcal{C}$=22). Our framework decreases communication costs by 36.8% and 16.41%, respectively. (a) Primary floorplan solution (|$\mathcal{C}$|=16). (b) Performance-aware floorplan solution (|$\mathcal{C}$|=16). (c) Primary floorplan solution (|$\mathcal{C}$|=22). (d) Performance-aware floorplan solution (|$\mathcal{C}$|=22).

The floorplan solutions are illustrated in Fig. 10. Fig. 10(a) and (c) shows the results of a 16-chiplet floorplan and a 22-chiplet floorplan without performance consideration. The width of the line indicates the data movement frequency between chiplets, and the length of the line indicates the communication latency between chiplets. The lines that represent low data movement frequency are omitted for clarity. Fig. 10(b) and (d) shows the results of the second stage of floorplan optimization, which obviously reduces the interchiplet communication cost. By reducing the HPWL wirelength by 0.57% and the communication cost between chiplets by 24.81%.

The performance-aware floorplan increases the overall area of the package by 0.86%. This is the tradeoff between the area cost and the performance improvement. The warpage also increases by 0.93% compared to the floorplan without performance. These overheads are acceptable considering the significant communication cost reduction and obvious performance enhancement by 13.18%.

Our framework improves the reliability of chiplet-based architecture by considering the warpage and bump stress issues in floorplan design and minimizing their effects. Moreover, the framework is compared with the methods that do not consider bump stress and warpage issues, and the result is shown in Table V.

In our framework, metrics like performance, cost, area, and reliability of chiplet-based architecture are considered. The experimental results demonstrate that ignoring performance

TABLE IV
COMPARISON OF THE METHODS WITH/WITHOUT PERFORMANCE METRIC

Methods	Testcases	HPWL (μm)	PA ($\times 10^5 \mu m^2$)	WPG ($\times 10^{-3} \mu m$)	ComCost ($\times 10^5$)	AveLat ($\times 10^9$ cycles)
ICCAD'22 [2]	$\mathcal{C}_8$	19743	1.114	0.116	1.269	6.05
	$\mathcal{C}_{16}$	107935	18.13	0.2898	4.565	81.99
	$\mathcal{C}_{22}$	376158	64.25	2.996	14.476	100.5
	$\mathcal{C}_{30}$	552483	115.1	7.065	37.37	738.9
Ours	$\mathcal{C}_8$	19918	1.126	0.1189	1.068	5.60
	$\mathcal{C}_{16}$	106021	18.14	0.2947	2.882	73.49
	$\mathcal{C}_{22}$	375430	65.44	3.019	12.10	83.6
	$\mathcal{C}_{30}$	545819	115.48	6.997	26.53	605.1
Ratio	$\mathcal{C}_8$	0.89%	1.14%	2.5%	-17.5%	-7.4%
	$\mathcal{C}_{16}$	-1.77%	0.1%	1.69%	-36.8%	-10.3%
	$\mathcal{C}_{22}$	-0.19%	1.85%	0.76%	-16.4%	-16.8%
	$\mathcal{C}_{30}$	-1.21%	0.35%	-1.25%	-28.4%	-18.1%
Average	N/A	**-0.57%**	0.86%	0.93%	**-24.81%**	**-13.18%**

TABLE V
ABLATION STUDY OF RELIABILITY CONSTRAINTS

Bump Constraints	Normalized Bump Stress	HPWL (μm)
Non-control	4.985	241016
Control	0.926	257571
Ratio	**-81.42%**	6.87%
Warpage Constraints	WPG ($\times 10^{-3} \mu m$)	PA ($\times 10^5 \mu m^2$)
Non-control	1.139	31.723
Control	1.115	31.613
Ratio	**-2.1%**	-0.347%

metrics in floorplan design will degrade performance. By incorporating performance factors into floorplan design early, the co-optimization of architecture and technology is realized.

V. DISCUSSION AND LIMITATION

A. Comparison With Traditional Floorplan Algorithms

Some previous works [21], [22] lack discussion of reliability issues in the 2.5-D package, i.e., the warpage threshold and avoiding the overlap between chiplets and the hotspot bumps. In our work, the constraint of the bump stress is expressed in (17) and it is shown in the lower left corner of Fig. 9. After introducing reliability in our framework, methods like the enumeration-based algorithm [21], the branch-and-bound algorithms [22], and other traditional algorithms like B* Tree [42], Sequence Pair [43], and Corner Block List [44] cannot fulfill the problem property. The main differences between our framework and some traditional algorithms lie in the complexity of the problem constraints.

Therefore, adopting MP methods in our framework can solve problems with multiple complex constraints well. The MP methods can take complex constraints like warpage threshold, bump reliability, interposer area, and wire length into consideration simultaneously during the iterative optimization. Meanwhile, the solutions to the MP problems can be finished by some excellent solvers like Gurobi [41]. Therefore, we compare our methods with MP-based methods in [2], and the results illustrate that our methods outperform previous work and achieve a better performance-aware floorplan solution.

The timing complexity cannot be analyzed precisely in the MP-solver-based optimization framework. That is because the optimization problems in our framework are solved by existing the MP solver Gurobi, so the timing complexity is highly reliant on the scale of the chiplets in the problem. On the contrary, the running time of the solver can be obtained for different tasks. For simple chiplet-based architecture like $\mathcal{C}_8$ and $\mathcal{C}_{16}$, the whole optimization process can be finished in 20 min to 1 h. For larger architecture like $\mathcal{C}_{22}$ and $\mathcal{C}_{30}$, to obtain a relative optimal solution will take 8–10 h. For some common chiplet-based architecture with 10–30 chiplets, our framework can give a good solution in an acceptable time compared to some traditional methods like the simulated annealing algorithm.

B. Limitation of the Framework

The thermal issue is important in 3-D IC or chiplet-based 2.5-D IC. The main reason is the thermal cannot be dissipated well with materials in the vertical direction. Chong et al. [11] have built the thermal model in their work to simulate the heat flow in 3-D IC. An accurate thermal model is constructed based on the actual physical parameters of the packaging. Our method focuses on introducing performance metrics into the floorplan stage to obtain an early optimization result, so the information related to thermal (e.g., voltage and accurate power) is not available in the current situation platform.

In the future, the framework will be developed continually by incorporating more metrics. For example, the thermal will be considered with a two-stage method in [23]. First, a thermal simulation model will be used to set the threshold, and after the first stage optimization, the *optChiplet* will continue to optimize the floorplan. Even though our current framework does not take into account the thermal effect, we hope the idea of combining reliability, technology, and performance can bring more insights into chip architecture into the IC manufacturing community.

VI. CONCLUSION

In this article, we present Floorplet, a performance-aware framework for co-optimizing the floorplan and performance of chiplet-based architecture. We address the challenges and

drawbacks of using chiplets for complex circuit systems, such as degraded performance due to interchiplet communication, reliability issues due to warpage and bump stress, and lack of realistic chiplet designs for analysis. We develop *parChiplet* to partition realistic SoC into functional chiplets, a simulation platform *simChiplet* to evaluate the performance impact of different floorplan solutions, and a floorplan framework *optChiplet* for chiplet-based architecture to consider reliability, performance, cost, and area metrics. We test our method on commonly used benchmarks and show its superiority over previous methods. Our work demonstrates the potential of using chiplets for designing high-performance and low-cost circuit systems. We hope that our framework can provide useful insights and guidance for future research and development of chiplet-based architecture.

REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Proc. IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998.

[2] Z. Zhuang, B. Yu, K.-Y. Chao, and T.-Y. Ho, "Multi-package co-design for chiplet integration," in *Proc. ICCAD*, 2022, pp. 1–9.

[3] D. C. H. Yu, "Advanced heterogeneous integration technology trend for cloud and edge," in *Proc. IEEE EDTM*, 2017, pp. 4–5.

[4] S. S. Iyer, "Heterogeneous integration for performance and scaling," *IEEE Trans. Compon., Packag. Manuf. Technol.*, vol. 6, no. 7, pp. 973–982, Jul. 2016.

[5] S. Pal, D. Petrisko, R. Kumar, and P. Gupta, "Design space exploration for chiplet-assembly-based processors," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 4, pp. 1062–1073, Apr. 2020.

[6] A. Sangiovanni-Vincentelli, Z. Liang, Z. Zhou, and J. Zhang, "Automated design of chiplets," in *Proc. ISPD*, 2023, pp. 1–8.

[7] D. Greenhill et al., "3.3 A 14nm 1GHz FPGA with 2.5 D transceiver integration," in *Proc. ISSCC*, 2017, pp. 54–55.

[8] Y.-K. Ho and Y.-W. Chang, "Multiple chip planning for chip-interposer codesign," in *Proc. 50th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, 2013, pp. 1–6.

[9] T. Hayashi, P. Y. Lin, R. Watanabe, and S. Ichikawa, "Development of highly reliable crack resistive build-up dielectric material with low Df characteristic for next-gen 2.5D packages," in *Proc. 71st ECTC*, 2021, pp. 570–576.

[10] F. Che, D. Ho, M. Z. Ding, and X. Zhang, "Modeling and design solutions to overcome warpage challenge for fan-out wafer level packaging (FO-WLP) technology," in *Proc. 17th EPTC*, 2015, pp. 1–8.

[11] S. C. Chong, S. S. B. Lim, W. W. Seit, T. C. Chai, and D. C. Sanchez, "Comprehensive study of thermal impact on warpage behaviour of FOWLP with different die to mold ratio," in *Proc. 71st ECTC*, 2021, pp. 1082–1087.

[12] M. Ahmad, J. DeLaCruz, and A. Ramamurthy, "Heterogeneous integration of chiplets: Cost and yield tradeoff analysis," in *Proc. 23rd Int. Conf. Therm., Mech. Multi-Phys. Simul. Exp. Microelectron. Microsyst. (EuroSimE)*, 2022, pp. 1–9.

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

[14] W. Haensch et al., "Silicon CMOS devices beyond scaling," *IBM J. Res. Develop.*, vol. 50, nos. 4-5, pp. 339–361, Jul. 2006.

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

[16] J. A. Cunningham, "The use and evaluation of yield models in integrated circuit manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 3, no. 2, pp. 60–71, May 1990.

[17] R. Irwin, K. Sahoo, S. Pal, and S. S. Iyer, "Flexible connectors and PCB segmentation for signaling and power delivery in wafer-scale systems," in *Proc. 71st ECTC*, 2021, pp. 507–513.

[18] M.-Y. Tsai and Y.-W. Wang, "A theoretical solution for thermal warpage of flip-chip packages," *IEEE Trans. Compon., Packag. Manuf. Technol.*, vol. 10, no. 1, pp. 72–78, Jan. 2020.

[19] M. Jung, D. Z. Pan, and S. K. Lim, "Chip/package co-analysis of thermo-mechanical stress and reliability in TSV-based 3D ICs," in *Proc. DAC*, 2012, pp. 317–326.

[20] K. Sakuma et al., "3D die-stack on substrate (3D-DSS) packaging technology and FEM analysis for 55μ-75μ mixed pitch interconnections on high density laminate," in *Proc. ECTC*, 2021, pp. 292–297.

[21] W.-H. Liu, M.-S. Chang, and T.-C. Wang, "Floorplanning and signal assignment for silicon interposer-based 3D ICs," in *Proc. DAC*, 2014, pp. 1–6.

[22] S. Osmolovskyi, J. Knechtel, I. L. Markov, and J. Lienig, "Optimal die placement for interposer-based 3D ICs," in *Proc. 23rd ASP-DAC*, 2018, pp. 513–520.

[23] F. Li et al., "GIA: A reusable general interposer architecture for agile chiplet integration," in *Proc. ICCAD*, 2022, pp. 1–9.

[24] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony, "2.2 AMD chiplet architecture for high-performance server and desktop products," in *Proc. ISSCC*, 2020, pp. 44–45.

[25] H. Liew et al., "Hammer: A modular and reusable physical design flow tool," in *Proc. 59th DAC*, 2022, pp. 1335–1338.

[26] 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.

[27] D. D. Sharma, G. Pasdast, Z. Qian, and K. Aygun, "Universal chiplet interconnect express (UCIe): An open industry standard for innovations with chiplets at package level," *IEEE Trans. Compon., Packag. Manuf. Technol.*, vol. 12, no. 9, pp. 1423–1431, Sep. 2022.

[28] A. Amid et al., "Chipyard: Integrated design, simulation, and implementation framework for custom SoCs," *IEEE Micro*, vol. 40, no. 4, pp. 10–21, Jul./Aug. 2020.

[29] K. Asanovic et al., "The rocket chip generator," Elect. Eng. Comput. Sci. Dept., Univ. California, Berkeley, CA, USA, Rep. UCB/EECS-2016-17, 2016.

[30] K. Asanovic, D. A. Patterson, and C. Celio, "The Berkeley out-of-order machine (BOOM): An industry-competitive, synthesizable, parameterized RISC-V processor," Elect. Eng. Comput. Sci. Dept., Univ. California, Berkeley, CA, USA, Rep. UCB/EECS-2015-167, 2015.

[31] H. Genc et al., "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *Proc. 58th DAC*, 2021, pp. 769–774.

[32] Y. Lee, C. Schmidt, A. Ou, A. Waterman, and K. Asanovic, "The Hwacha vector-fetch architecture manual, version 3.8. 1," Elect. Eng. Comput. Sci. Dept., Univ. California, Berkeley, CA, USA, Rep. UCB/EECS-2015-262, 2015.

[33] V. Vashishtha, M. Vangala, and L. T. Clark, "ASAP7 predictive design kit development and cell design technology co-optimization," in *Proc. ICCAD*, 2017, pp. 992–998.

[34] J. -W. Jang et al., "Sparsity-aware and re-configurable NPU architecture for samsung flagship mobile SoC," in *Proc. 48th ISCA*, 2021, pp. 15–28.

[35] S. R. Sridhara et al., "Microwatt embedded processor platform for medical system-on-chip applications," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 721–730, Apr. 2011.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[37] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[38] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *ACM SIGARCH Comput. Architect. News*, vol. 34, no. 4, pp. 1–17, 2006.

[39] H. J. Curnow and B. A. Wichmann, "A synthetic benchmark," *Comput. J.*, vol. 19, no. 1, pp. 43–49, 1976.

[40] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, 2017, pp. 1–11.

[41] "Gurobi optimizer." Accessed: Mar. 21, 2023. [Online]. Available: http://\penalty\z@{}www.\penalty\z@{}gurobi.\penalty\z@{}com/\penalty\z@{}.\penalty\z@{}

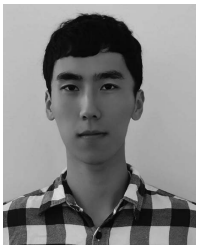
[42] Y.-C. Chang, Y.-W. Chang, G.-M. Wu, and S.-W. Wu, "B*-trees: A new representation for non-slicing floorplans," in *Proc. 37th DAC*, 2000, pp. 458–463.

[43] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "Rectangle-packing-based module placement," in *Proc. ICCAD*, 1995, pp. 472–479.

[44] X. Hong et al., "Corner block list: An effective and efficient topological representation of non-slicing floorplan," in *Proc. ICCAD*, 2000, pp. 8–12.

Shixin Chen received the B.Eng. degree in VLSI design and system integration from Nanjing University, Nanjing, China, in 2022. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. B. Yu.

His research interests include agile hardware design, hardware design space exploration, chiplet integration, and machine learning in electronic design automation.



Shanyi Li received the B.Eng. degree from Tsinghua University, Beijing, China, in 2019. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. T.-Y. Ho.

His current research interest is electronic design automation (EDA), especially EDA for 2.5-D/3-D packaging.

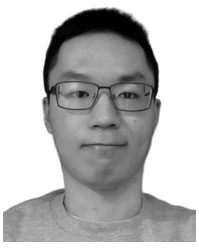
Zhen Zhuang received the B.Eng. and M.Eng. degrees from Fuzhou University, Fuzhou, China, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree with The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. T.-Y. Ho.

His current research interest is electronic design automation (EDA), especially EDA for 2.5-D/3-D packaging.

Mr. Zhuang was a recipient of three ICCAD/ISPD contest awards.

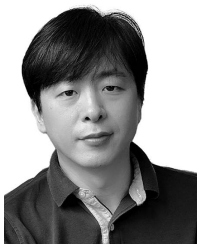
Su Zheng received the B.Eng. and M.S. degrees from Fudan University, Shanghai, China, in 2019 and 2022, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, under the supervision of Prof. B. Yu and Prof. M. D. F. Wong.

His research interest is to solve critical problems in electronic design automation with advanced artificial intelligence methods.



Zheng Liang (Graduate Student Member, IEEE) received the B.S. degree from EECS, Peking University, Beijing, China, in 2020. He is currently pursuing the Ph.D. degree with the Department of EECS, University of California at Berkeley, Berkeley, CA, USA, under the supervision of Prof. A. Sangiovanni-Vincentelli.

His research interests include optimization algorithms and corresponding applications in electronic design automation, computer systems, and machine learning.



Tsung-Yi Ho received the Ph.D. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2005.

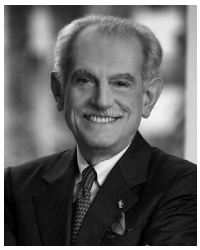
He is currently a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Prof. Ho was a recipient of the Best Paper Award at IEEE TCAD in 2015. He currently serves as the VP Conference of IEEE CEDA, and the Executive Committee of ASP-DAC and ICCAD. He is a Distinguished Member of ACM.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received ten Best Paper Awards from IEEE TSM 2022, DATE 2022, ICCAD 2021 and 2013, ASPDAC 2021 and 2012, ICTAI 2019, *Integration, the VLSI Journal* in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, and six ICCAD/ISPD contest awards. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is an Editor of IEEE TCCPS Newsletter.



Alberto L. Sangiovanni-Vincentelli received the Dottore in Ingegneria degree (summa cum laude) in electrical engineering and computer science from the Politecnico di Milano, Milan, Italy, in 1971, and four Honorary Doctorate degrees from the University of Aalborg, Aalborg, Denmark, in 2009; the KTH Royal Institute of Technology, Stockholm, Sweden, in 2012; the AGH University of Science and Technology, Kraków, Poland, in 2022; and the University of Rome "Tor Vergata," Rome, Italy, in 2022.

He is the Edgar L. and Harold H. Buttner Chair of Electrical Engineering and Computer Sciences with the University of California at Berkeley, Berkeley, CA, USA. He is an author or coauthor of over 1000 papers, 17 books, and three patents in design tools and methodologies, large-scale systems, embedded systems, hybrid systems, and AI. He was a Co-Founder of Cadence, San Jose, CA, USA, and Synopsys, Sunnyvale, CA, USA, the two leading companies in Electronic Design Automation. He was a Consultant or a member of the Advisory Boards of several companies, such as BMW, Mercedes, Magneti Marelli, Intel, ST microelectronics, HP, General Motors, United Technologies, Lutron, Lendlease, and Elettronica. He is currently a member of the following boards of directors: Cadence, KPIT Technologies, eGap, Exein, and Cy4Gate. He is a Chairman of the Board of Quantum Motion, Innatera, Phoelex, e4Life, and Phononic Vibes. He was a member of the Scientific Council of the Italian National Science Foundation (CNR) from 2001 to 2015. From February 2010 to December 2020, he was a member of the Executive Committee of the Italian Institute of Technology, where he is currently a member of the Technical Scientific Committee. In September 2023, he has been appointed as the President of the Chips.it, the 250M Euro Foundation of the Italian Government to foster integrated circuit design. He is the Chairperson of the Strategy Board and the International Advisory Board of the Milano Innovation District (MIND).

Dr. Sangiovanni-Vincentelli is the recipient of several academic honors, and research awards, including the IEEE/RSE Wolfson James Clerk Maxwell Medal "for groundbreaking contributions that have had an exceptional impact on the development of electronics and electrical engineering or related fields" and the BBVA Frontiers of Knowledge Award in the Information and Communication Technologies category: "for transforming chip design from a handcrafted process to the automated industry that power today's electronic devices." He is an ACM Fellow and a member of the National Academy of Engineering.