Automated Design of Chiplets

Alberto Sangiovanni-Vincentelli
Zheng Liang
alberto@berkeley.edu
zhliang@berkeley.edu
University of California, Berkeley
Berkeley, California, USA

Zhe Zhou
Jiaxi Zhang
zhou.zhe@pku.edu.cn
zhangjiaxi@pku.edu.cn
Peking University
Haidian, Beijing, China

ABSTRACT

Chiplet-based designs have gained recognition as a promising alternative to monolithic SoCs due to their lower manufacturing costs, improved re-usability, and optimized technology specialization. Despite progress made in various related domains, the design of chiplets remains largely reliant on manual processes. In this paper, we provide an examination of the historical evolution of chiplets, encompassing a review of crucial design considerations and a synopsis of recent advancements in relevant fields. Further, we identify and examine the opportunities and challenges in the automated design of chiplets. To further demonstrate the potential of this nascent area, we present a novel task that showcases the promising future of automated chiplet design.

CCS CONCEPTS

• **Hardware → 3D integrated circuits**; **Platform-based design**; **System on a chip**; **Hard and soft IP**.

KEYWORDS

chiplet, SoC, design automation, physical design, consolidation

ACM Reference Format:
Alberto Sangiovanni-Vincentelli, Zheng Liang, Zhe Zhou, and Jiaxi Zhang. 2023. Automated Design of Chiplets . In *Proceedings of the 2023 International Symposium on Physical Design (ISPD '23), March 26–29, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3569052.3578917

1 INTRODUCTION

As technology advances and the feature size of logic transistors decreases, the function density increases, with billions of transistors now contained within a single die. The higher transistor count presents a significant challenge to traditional design tools and flows. Block-based (a.k.a., IP-based) design is intended to offer a solution to the complexity problem by dividing a system into smaller, reusable blocks, and connecting them through a Network-on-Chip (NoC) communication scheme [3]. The resulting chip is called System-on-Chip (SoC) given the similarity of the approach followed in PCB systems. Yet, manufacturing large monolithic SoCs has become increasingly challenging as advanced technology nodes enter into production. SoCs are not only more prone to fabrication defects

ISPD '23, March 26–29, 2023, Virtual Event, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9978-4/23/03.
https://doi.org/10.1145/3569052.3578917

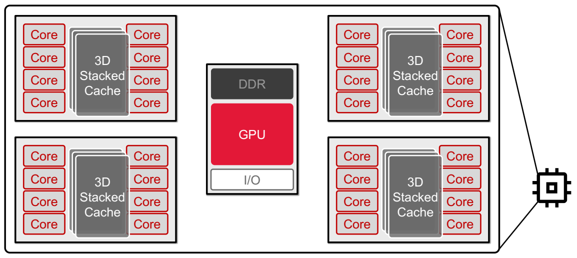


Figure 1: A conceptional 64-core processor based on three types of chiplets, core complex die (CCD), IO die (IOD), and stacked 3D cache (adapted from a family of products [33]).

due to their die sizes and to the need of using finer lithography patterns, resulting in higher recurring engineering (RE) costs, but also, because of the inflexibility of tightly integrated components, they require distinct tape-outs for different products even if several blocks are shared, causing non-recurrent engineering (NRE) costs to skyrocket. To overcome these problems, a methodology that considers the possibility of decomposing an SoC into multiple chips (a.k.a. *chiplets*) that are then packaged together in the same substrate emerged. The design steps are as follows: 1) identify a set of blocks in the original SoC design that will be manufactured separately with the best technology for each of them in terms of manufacturing costs (yield, mask set cost, processing). 2) once manufactured as separate chips, select known good dice (KGDs), and 3) package them in System-in-Packages (SiPs) with advanced packaging technologies to improve performance versus packaging each chip separately. The advantages of a chiplet-based solution can be summarized as 1) increased yield, 2) decreased integration complexity, 3) possibility of exploiting different nodes and different technologies for different chiplets, and 4) increased re-usability of *pre-manufactured* chiplets versus IPs that need to be manufactured every time a new SoC is designed. It has been estimated that four 10×10 mm^2 dice yield 30% more good dice than a single 20×20 mm^2 die.

In this paper, we present a review of the crucial aspects of chiplet design and the advancements in design automation for chiplets. We believe that the automated design of chiplets must prioritize cost reduction, performance optimization, and, most significantly, re-usability across various products. We observe that existing EDA tools fall short in addressing these challenges, particularly with respect to identifying chiplets that can be re-used across a family of products. Motivated by this finding, we also present a design method for this task. Our findings suggest that this is a promising area for future research and development efforts.

2 BACKGROUND

In this section, we review the history of chiplets and related concepts and identify the advantages and disadvantages of chiplet-based design.

2.1 A Brief History

The concept of systems assembled using "naked" chips (another way to represent chiplets) in multi-chip modules (MCMs) is not new. However, motivations and objectives have evolved over the years to make this approach increasingly appealing as IC manufacturing technology progresses. Many electronic systems used MCMs decades ago when performances in high-end devices could not be met with packaged chips on PCBs while a single chip would not be feasible due to the number and complexity of all components. Further, add-on components designed as functional patches or differentiating features were manufactured separately as chiplets for modularity and re-usability. For example, before the 2000s, large SRAM caches in CPU processors were implemented as chiplets like those in IBM Power-1 [2] and Intel Pentium Pro [12].

However, in the 2010s, as Moore's law kept driving the growth of SoCs, it became easier to directly pack more components into a single chip, thus solving the performance problem, and the discussion of multi-die systems faded away. In recent years, the concept of chiplet has become popular again due to factors such as the cost of large monolithic chips, the different technology needs of the IPs used in the design (e.g., digital logic, analog parts, MEMS and power units), the NRE barrier, and the inevitable slowing of Moore's law. In 2015, Marvell announced its Modular Chip (MoChi) [40] architecture that broke SoCs into LEGO-like standard building blocks. In 2017, AMD announced its MCM-based Zen CPU architecture [26, 39] (Figure 1). Other companies also quickly embraced this approach [15, 45]. Recently, the IEEE Electronics Packaging Society (EPS) formed a new working group [13] on SiP; semiconductor companies are discussing specifications such as the Universal Chiplet Interconnect Express (UCIe) [10].

Interestingly, memory circuits like DRAM can be taken as special cases of chiplets. In the beginning, manufacturing was similar for memory and logic devices. In fact, DRAM can be fabricated as embedded DRAM (eDRAM) [32, 43] using logic-compatible technologies, but there are also separate eDRAM devices and CPU dice in the same product package for some systems [17]. In recent years, High Bandwidth Memory (HBM) for high-performance computing (HPC) markets [14, 31] has been a driving factor for advanced 2.5D packaging technologies with chiplets. Stacked DRAM chips are usually integrated with logic chips with silicon interposers in the same package. The recently announced AMD Instinct MI300 Data Center Accelerated Processor Unit (APU) [1] has a total of 13 chiplets, most of them 3D-stacked, to create a SiP with twenty-four CPU cores with a graphics engine and eight stacks of HBM3.

To witness the strong progression of chiplets, Yole Group forecasts the chiplet-based semiconductor market to be over $205B by 2032. Samsung Foundry estimated that over 50% of advanced-node designs are chiplet-based.

[1]Estimated NRE cost source: Handel Jones, IBS, 2018; estimated cost per transistor source: RetiredEngineer, Twitter, 2020

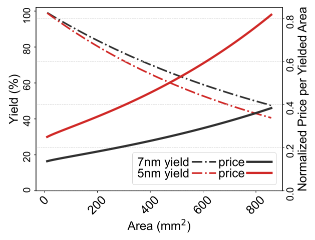


Figure 2: **Yield and price per yielded area**

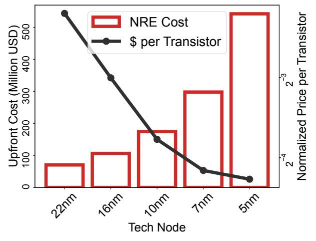


Figure 3: **NRE and RE cost trends** [1]

2.2 Advantages

• Lower Manufacturing Costs

The use of chiplets in designs can significantly lower manufacturing costs through improved yields. A commonly used yield model for chips is the negative binomial model [9] below, which considers the yield Y has a function of die area s and two parameters: defect density d_0 and cluster parameter α.

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

For 7 nm technology in 2020, typical values for d_0 and α were 0.09 cm^{-2} and 10, respectively. We can estimate the manufacturing costs using the processed wafer's yield Y and unit price P_0.

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

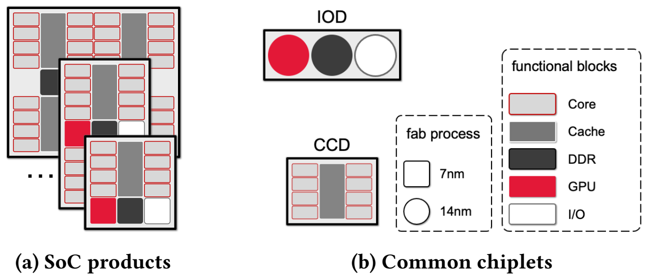
Equation (2) used the Taylor expansion to demonstrate that $\tilde{p}(s)$, manufacturing cost per yielded area, rises quickly with larger die sizes (also shown in Figure 2). By breaking down a monolithic SoC into smaller chiplets, we can lower the cost per silicon area and, ultimately, lower manufacturing costs.

• Better Reusability and Modularity

Chiplets enable lower NRE costs and shorter time to market in designing products for different markets by mixing and matching modular components. By combining different chiplets, companies can create new system designs without having to start from scratch in designing, testing, and manufacturing individual SoC chips.

The industry has faced the persistent rise of NRE costs, and SoCs worsen the situation. Such concern dates back to before the 2000s [42]. In an effort to mitigate this challenge, cache chiplets were introduced in the Pentium Pro series processors [12]. These chiplets were designed to support a diverse range of products that required varying cache capacities. In 2006, pioneers [38] from Philips proposed using "companion chips" to encapsulate differentiating functionalities in high-end television systems. They demonstrated that *"splitting a single complex application-specific standard product (ASSP) into a smaller main chip with (multiple) companion chips is advantageous."*

As chip fabrication becomes increasingly complex, NRE cost barriers become higher for advanced technologies. According to a widely accepted estimation in 2018, shown in Figure 3, the price per transistor remains low. To offset the upfront cost, vendors need to produce and sell large volumes of products, with previous research from Arm [46] suggesting a minimum of 10 million units per product for advanced nodes. However, this is a challenging target for most products. An alternative solution is to use a mix-and-match approach with common chiplets across different products, which helps consolidate the design and manufacture of chips.



(a) SoC products **(b) Common chiplets**

Figure 4: Reusable chiplets as an alternative to distinct SoCs

The recent AMD Zen architecture exemplifies this approach, as illustrated in Figure 4. Server and desktop processors with varying core counts may share the identical core complex die (CCD) and IO die (IOD) design without modification. The mix-and-match approach enables the creation of a broad range of products by flexibly composing only a few kinds of pre-manufactured chiplets (Figure 4b). In contrast, monolithic SoCs (Figure 4a) require separate validation and taping out for each product, incurring higher NRE costs and longer time-to-market.

• **Optimization via Heterogeneous Technologies**

Further optimization can be achieved through the use of heterogeneous manufacturing technologies for chiplets, known as heterogeneous integration(HI) [20]. Different components of a system have varying requirements for these technologies. Typically, designers must use a single technology to manufacture a monolithic die for the entire system. However, designers can select the most appropriate technology for each component by decomposing the system into multiple dies.

The I/O interfaces of processors are an example. Analog circuits dominate I/O interfaces and do not benefit as much from smaller technology nodes as logic devices do. Using a less advanced technology node for a separate I/O die can significantly reduce costs compared to using the most advanced technology node. The AMD Zen 2 architecture (Figure 4) utilizes this approach. The CCD containing the CPU cores and caches is manufactured with 7nm technology, while the I/O die is manufactured with 14nm technology, resulting in up to 50% reduction in manufacturing costs with a minimal performance impact [16].

In addition to utilizing different technology nodes, designers can also take advantage of specialized processes with different materials and electron devices for specific components. [20] showed that heterogeneous chiplets manufactured using conventional CMOS and SiGe BiCMOS could provide superior performance. Indeed, various memory devices like DRAM gained substantial growth after customized device structures and materials were employed. This success cannot be achieved with fully-integrated SoCs with identical fabrication processes.

2.3 Concerns

Despite the benefits of chiplets, it is essential to keep in mind their drawbacks. To avoid the pitfall of constructing systems made up of tiny components like individual transistors, the indiscriminate use of chiplets should be avoided.

Table 1: Comparison of different interconnections

	Intra-die	Inter-die	PCB
Latency (ns)	< 0.5	$10 - 100$	> 100
Bandwidth (GiB/s)	$> 10^4$	$100 - 1000$	$2 - 100$
Power Consumption	Low	Medium	High
Material Cost	Low	High	Medium

Chiplets often result in lower communication performance, with higher latency, reduced bandwidth, and increased power consumption in inter-die connections (as shown in Table 1). Improper design can result in excessive communication needs and significant performance loss.

In addition, chiplets incur extra packaging costs and die-to-die interfaces, which may offset RE cost savings. Research shows that for highly defective advanced technology nodes, smaller chiplets may be cost-effective, but breaking down a 14nm SoC into more than five chiplets raises manufacturing costs, making chiplets less favorable than monolithic SoCs.

Lastly, if proper reuse is not considered, chiplet-based designs can result in higher NRE costs. Each distinct chiplet requires its own masks, and a single tape-out can cost 100 million US dollars for the most advanced technology. Decomposing a product into multiple chiplets might result in dissimilar chiplets, raising total NRE costs. However, NRE costs can be reduced if chiplets can be reused among a product family, as discussed in previous sections.

3 OPPORTUNITIES AND CHALLENGES

Advances in physical implementations have laid the foundation for the success of chiplets in applications, and their pervasive application provided massive use cases in the research on physical design for chiplets. In this section, we summarize the progress in chiplet-related design automation and share our view of unresolved challenges and corresponding opportunities.

3.1 Physical Implementation

Chiplets are conventional dice, and hence, traditional backend implementation tools apply to chiplets. However, the implementation of chiplets involves the integration of multiple dice within a single package, which necessitates close alignment with the design of the package. To accommodate this requirement, design environments like Cadence Integrity 3D-IC [41] with seamless chiplet, package, and PCB integration are necessary.

Furthermore, considerable progress has been made in the physical implementation of chiplets in MCM/2.5D/3D packages. Several studies, such as those by Jung et al. [25] and Zhuang et al. [48], have explored the joint thermo-mechanical analysis of chiplets and packages. Previous works ([19, 29, 34, 48]) have conducted research on the placement and floor planning of chiplets in packages with different methods and considerations. Chiang et al. [7] have presented a unified method for routing redistribution layers in both chiplets and interposers. The field of the physical implementation of chiplets incorporating multiphysics models holds great promise.

As noted by Jiang et al. [21–23], there are numerous opportunities for improvement in areas such as placement and routing,

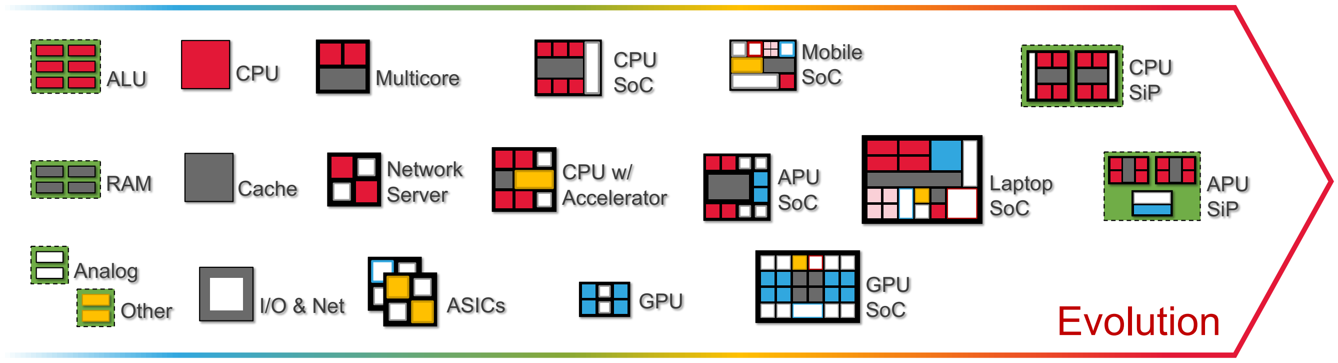


Figure 5: Evolution of chip design: from small-scale components to SoCs and SiPs

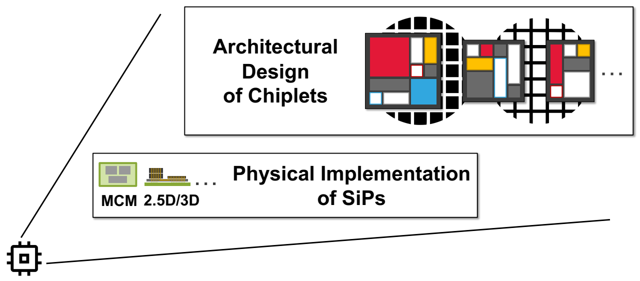

Architectural Design of Chiplets
Physical Implementation of SiPs
MCM
2.5D/3D

Figure 6: Automated design of chiplets

timing analysis, electrical analysis, thermo-mechanical analysis, and testing in the context of chiplets and die-package co-design. We conclude that physical-implementation design automation of chiplets is on track and will keep thriving.

3.2 Architectural Design

Advances in chiplet physical implementation have been made. However, all existing tools cannot handle cases where the architectural design (Figure 6), a determining factor in the quality of results, has not been fully determined. To reduce manufacturing costs, chiplets should be used across a family of products. To retain and even enhance system performance, designers must strategically choose components for each chiplet and determine appropriate manufacturing technology.

Many designers confirmed the merits of chiplets. Several papers [15, 33, 40, 45] proposed to develop a chiplet platform for different products. However, manual architectural design is usually arbitrary and lacks justification for optimality. Different chiplet architectures cannot be easily evaluated, especially when measuring reusability. And most metrics focus on the results of a single product instead of the planning of the whole family of products.

Design choices in multi-die systems are subject to change based on evolving manufacturing technologies, leading to intricate modifications and huge engineering efforts. The historical evolution (Figure 5) of these systems exhibits a back-and-forth pattern. Prior to the 2010s, most DRAM controllers in personal computers were integrated into separate northbridge [8, 30, 35] chips located on motherboards, whereas in subsequent years, they were integrated into the CPU chip. However, the recent prevalence of chiplet design has resulted in vendors returning to using separate dice similar to previous designs, such as the IO die in AMD Zen. The integration [6, 30] and disintegration [2, 12, 44] of SRAM caches have also undergone a similar evolution over the years. Given these historical patterns, it is imperative that lessons should be learned and that a unified chiplet design automation platform should be developed for the automatic decision-making of functional block aggregation and disaggregation in multi-die systems in order to achieve engineering efficiency and avoid past mistakes.

The future of the semiconductor industry heavily relies on system-level design optimization, as direct scaling of transistors becomes challenging. Instead of relying on general-purpose processors, heterogeneous computing and domain-specific architecture are promoted as more efficient solutions through specialization [18]. Companies are increasingly designing custom hardware, such as deep-learning accelerators [5, 24, 28] and data processing units (DPU) [4, 11], and integrating them with software stacks. To minimize hardware fragmentation and NRE costs, developers should consider using platform-based design [36] with chiplets. Cloud computing and warehouse-scale computers add new hierarchies to electronic system design and require custom hardware acceleration. Advanced packaging [47] offers opportunities for scaling at the data-center level by integrating more components into SiPs. The resurgence of chiplets allows for resource aggregation and disaggregation at various scales. All these trends indicate the necessity of automated architectural design of chiplets.

Despite being promising, the field of automated architecture design remains underdeveloped. We believe that this field is worth attention. To examine its potential, we formulated a new task: generate common chiplets for a family of products.

4 CHIPLET CONSOLIDATION

Chiplet consolidation is an essential step in identifying the most convenient decomposition of complex SoCs into chiplets. This is done considering the cost and performance of the resulting system. The goal of chiplet consolidation is to generate the high-level structures of common chiplets for a family of products/SoCs given the system block diagrams, such as task graphs of all products, so that we can reduce the RE and NRE costs while maintaining system performance. It serves as a plug-in extension of traditional design flows. The outputs can be fed into physical implementation tools as if the consolidated chiplets are the original products/SoCs.

4.1 Modeling

We model the block diagram of n input SoCs as weighted graphs $G^{(1)}, \ldots, G^{(n)}$. Each node is an IP block from a shared library with r types of IP resources, and edges are communications between them. Node weights represent the area of different blocks; edge weights are interconnection bandwidths. Every node comes with a label indicating the IP type. Our problem is finding a set of **template graphs** $T_1, \cdots, T_t$ so that we can manufacture them as common chiplet IPs and find a specific way to "mix and match" concrete chiplets *as instances of these templates* to "emulate" the function of input SoCs. Table 2 contains notations for reference.

Table 2: Common notations

Notation	Description
$G^{(i)}$	i-th input graph as a product diagram
$q^{(i)}$	product quantity weight of $G^{(i)}$
n	Number of input graphs
V	Vertex set of a graph G
E	Edge set of a graph G
A	Adjacency matrix of a graph G
L	Laplacian matrix of a graph G
l	Number of nodes in a graph G
$G_j^{(i)}$	j-th sub-graph of i-th input graph
$m^{(i)}$	Number of sub-graphs in $G^{(i)}$
C	An edge cut set between two sub-graphs
x	Indicator vector of a vertex set
T_k	k-th template graph as a common chiplet
t	Number of template graphs
s_k	Estimated area of T_k
z	Template assignment vector of an instance
R	IP resource matrix
r	r types of IP resources

4.2 Objectives

In this subsection, we introduce three major metrics, total NRE cost, total RE cost, and inter-die communication, to evaluate the results.

- **Total NRE Cost:** We simply use the value of t to measure the total NRE cost. The more chiplets we need to manufacture, the higher the NRE cost we need to take.
- **Total RE Cost:** We use φ_k to represent the RE cost of each template chiplet T_k. Each φ_k will be estimated using the area s_k of T_k and the cost model Equation (2) in Section 2.2. The total RE cost φ can be calculated as the quantity-weighted sum of the RE cost $\varphi_j^{(i)}$ of each constituting chiplet instance in the product.
- **Inter-die Communication:** We use $\rho^{(i)}$, the cut size of the partitioning of each graph, to represent the inter-die communication bandwidth requirement of $G^{(i)}$ when implemented by chiplets. The corresponding quantity-weighted sum ρ will be used as the final metric.

4.3 Approach

We use a "Partition-Merge" approach to generate valid solutions, as depicted in Figure 7. This approach involves partitioning the input graphs into sub-graphs that adhere to size constraints, followed by merging similar sub-graphs into a limited number of template graphs. Upon fabrication, these template graphs, representing common chiplets, are assembled in accordance with the partitioning of input system diagrams. In this process, we introduce unused blocks in each concrete use case to trade silicon area (RE costs) for better chiplet reusability (NRE costs).

In order to establish the objectives with a rigorous analytical approach, we go through the partitioning and merging procedures, which allow us to derive the evaluation metrics analytically. Without loss of generality, we initially adopted a simplified approach using a single input graph devoid of superscripts. Upon completion of this demonstration, our analysis will be expanded to include n input graphs and the incorporation of superscript notations.

• Partition.

For a graph $G = (V, E)$ [2]of size l with adjacency matrix A and node weight w, the cut set C between two disjoint vertex subsets V_1, V_2 or their induced sub-graphs

$$G_1(V_1, E_1) = G[V_1], \quad G_2(V_2, E_2) = G[V_2]$$

, is

$$C = \{(v_1, v_2) \in E \mid v_1 \in V_1, v_2 \in V_2\} \tag{3}$$

, and its size is

$$|C| = \sum_{(v_i, v_j) \in C} A[i, j] \tag{4}$$

In a traditional graph partition problem, we partition the vertex set V into m disjoint subsets $V_1, \ldots, V_m$ and try to minimize the total cut size subject to the sub-graph size constraint (Equation (5)).

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} \sum_{j=i+1}^{m} |C_{i,j}| \\ \text{s.t.} \quad & \underline{\omega} \le |V_i| \le \overline{\omega}, \quad \forall i \in \{1, \ldots, m\} \end{aligned} \tag{5}$$

$|V_i| = \sum_{v_k \in V_i} w[k]$ is the size of V_i, and $\underline{\omega}, \overline{\omega}$ are predefined bounds.

According to the spectral graph theory [37], the cut size of a graph bi-partition problem (consider a vertex subset V_i and its complement set $\overline{V_i}$) can be derived analytically using V_i's indicator vector and the graph Laplacian matrix L of the original graph. We can rewrite the total cut size and constraint in the graph partition problem as follows.

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{j=1}^{m} x_j^T L x_j \\ \text{s.t.} \quad & \underline{\omega} \le w^T x_j \le \overline{\omega}, \quad \forall j \in \{1, \ldots, m\} \end{aligned} \tag{6}$$

• Merge.

[2]For many cases in this paper, we will focus on each input, like G, and the corresponding variables, like G_j, for clarity without using the superscript unless otherwise specified. Also, we have to reuse indices like i, j, k, and in different sections, they may be used to refer to different quantities.

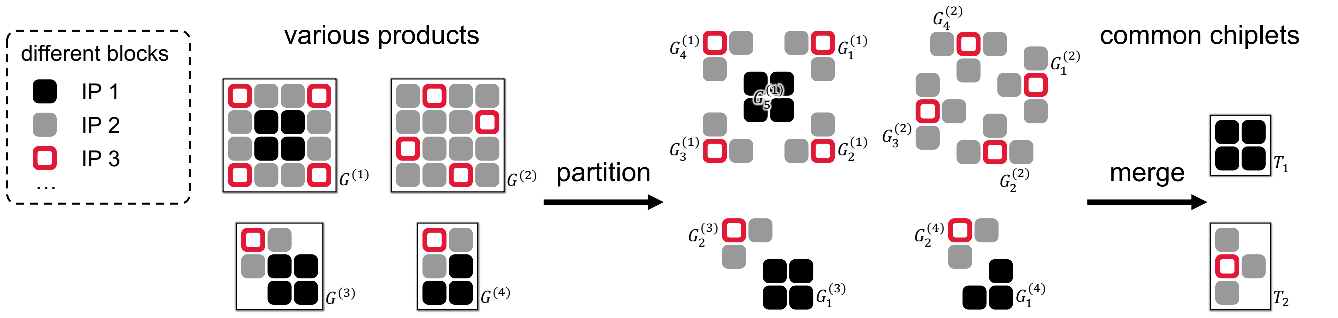


Figure 7: Our Partition-Merge scheme. Various products represented as input graphs ($G^{(1)} - G^{(4)}$) are partitioned into sub-graphs. Similar sub-graphs are merged into a limited number of template graphs (e.g., T_1,T_2) for fabrication.

In this step, we merge m different sub-graphs into t templates ($t \ll m$), reducing the number of distinct chiplets. Given a sub-graph G_j, we need to determine which type k it is. We use a one-hot vector $z_j \in \{0,1\}^t$ to represent the assignment of templates.

$$z_j[k] = \begin{cases} 1, & \text{if } G_j \text{ is an instance of } T_k \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

We can also use a matrix $Z = [z_1, \ldots, z_m] \in \{0,1\}^{t\times m}$ to pack the assignment column vectors, and each row $Z[k,:]$ represents all sub-graphs implemented as instances of T_k.

To construct a template graph T_k, we need to collect *sufficient resources.* With r unique IP resources and area weight $\vec{s_0} \in \mathbb{R}^r$, the resource matrix of the input graph G of size l is $R_0 \in \{0,1\}^{r\times l}$

$$R_0[i,j] = \begin{cases} 1, & \text{if } v_j \text{ is of type } i \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

, and the aggregated resource matrix of different sub-graphs $\{G_1, \ldots, G_m\}$ is $R \in \mathbb{R}^{r\times m}$, representing how many resources of type i each sub-graph G_j requires.

$$R[i,j] = \kappa, \text{ if } \kappa \text{ nodes of type } i \text{ are in } G_j \tag{9}$$

We can obtain it by aggregating R_0 and $X = [x_1, \ldots, x_m]$, the packed form of sub-graph indicators in the previous partitioning process.

$$R = R_0 X \tag{10}$$

Similarly, we denote the resource matrix of templates as $\overline{R} \in \mathbb{R}^{r\times t}$.

$$\overline{R}[i,k] = \kappa, \text{ if } \kappa \text{ nodes of type } i \text{ are in } T_k \tag{11}$$

The "sufficient resource" constraint can be expressed as follows.

$$\overline{R}[i,k] = \max_{j\in\{1,\ldots,m\}} \{R[i,j] \cdot Z[k,j]\} \tag{12}$$

In this way, we can construct each template graph T_k by assigning and connecting nodes from corresponding G_j.

• **Solve.**

Now, we will show how to formulate the metrics analytically and obtain a valid solution from the templates $\{T_1, \ldots, T_t\}$ we constructed above.

The goal is to calculate the evaluation metrics ρ and φ defined in Section 4.2.

Note that $\rho^{(i)}$ can be computed by the cut sizes between $V_1^i, \ldots, V_{m^{(i)}}^{(i)}$ using Equation (6) as follows

$$\rho^{(i)} = \frac{1}{2}\sum_{j\neq k}\left|C_{j,k}^{(i)}\right| = \frac{1}{2}\sum_{j=1}^{m^{(i)}} {x_j^{(i)}}^T L^{(i)} x_j^{(i)} \tag{13}$$

The area s_k of each chiplet template T_k, and can be estimated by

$$s_k = |T_k| = \vec{s_0}^T \overline{R}[:,k] \tag{14}$$

. The corresponding RE costs of template chiplets can be estimated by Equation (2) and are accumulated to get φ.

Given **each fixed t**, we have two optimization objectives ρ and φ to be minimized. We optimize a weighted sum

$$J_\theta = \theta\rho + (1-\theta)\varphi, \quad \theta \in [0,1] \tag{15}$$

with different values of θ to explore the trade-off between these two objectives. By enumerating t, we can explore the entire design space with different NRE cost budgets. In practice, we first obtain the empirical minimums of ρ and φ by letting $\theta = 1, 0$, and then minimize the normalized objectives. We implemented our algorithm by mathematical programming using Gurobi.

4.4 Evaluation

We construct a family of heterogeneous SoCs denoted by $G^{(1)}$, $G^{(2)}$, $G^{(3)}$, and $G^{(4)}$. These SoCs contain three IP blocks: IP-1, IP-2, and IP-3. The estimated areas and corresponding RE costs of chiplets and SoCs are listed in Table 3.

Table 3: **Estimated area and cost breakdown of dies (7nm technology)**

	IP-1	IP-2	IP-3	$G^{(1)}$	$G^{(2)}$	$G^{(3)}$	$G^{(4)}$	T_1	T_2
# IP-1	1	0	0	22	1	1	13	4	1
# IP-2	0	1	0	16	0	4	0	4	0
# IP-3	0	0	1	15	20	49	47	4	7
Area (mm^2)	7	13	12	542	247	647	655	128	91
RE cost ($)	N/A	N/A	N/A	115	41	151	154	19	13

We compare the generated common chiplets with monolithic SoCs to demonstrate the cost savings of our approach. With parameters in Table 4 and each $m^{(i)}$ determined by the SoC area divided by $\underline{\omega}$, our algorithm can generate two chiplets, T_1 and T_2. Their configurations are listed in Table 3.

Table 4: Experiment settings

t	θ	$q^{(i)}$	$\underline{\omega}$ (mm^2)	$\overline{\omega}$ (mm^2)	Run Time (min)	Threads
2	0.2	1	80	200	30	16

When implemented using chiplets T_1 and T_2, $G^{(1)}$ consists of 6 T_1 chiplets, and the RE cost per product is \$114; $G^{(2)}$ consists of 3 T_2 chiplets, and the RE cost per product is \$39; $G^{(3)}$ consists of 1 T_1 chiplet and 7 T_2 chiplets, and the RE cost per product is \$110; $G^{(4)}$ consists of 2 T_1 chiplets and 6 T_2 chiplets, and the RE cost per product is \$116. Using chiplets can save up to 28% on RE costs for specific products compared to monolithic SoCs. More importantly, we can achieve the functionality of 4 different SoCs with only two chiplets, cutting the NRE cost by about 50%. This advantage can benefit many products with moderate market volumes. For example, assuming that the NRE cost is roughly proportional to t, and for each tape-out, the NRE cost is about \$20M and that each of the four products has 10,000 shipments, the total cost will be approximately \$44M if using the two common chiplets. In contrast, the total cost will be \$84M if using monolithic SoCs, almost doubling the cost.

5 DISCUSSIONS

Despite encouraging results, our model and solution have limitations. First, the construction method only explores a subset of the design space, potentially sub-optimizing the Quality of Results (QoR). Second, our synthetic inputs may not accurately reflect real-world applications. To thoroughly evaluate this task, a benchmark consisting of systems based on shared building blocks is needed. Currently, such a benchmark is not publicly available; thus, we call for contributions to establish a standard benchmark. Last, our approximate model and high-level architecture results have not been tested in realistic circuit designs. In future work, we plan to connect the results to downstream physical design tools for more accurate assessments.

Our simplified task cannot represent the entire landscape of chiplet design automation. There are still many opportunities and challenges. We list some potential directions below.

5.1 Heterogeneous Integration

In our task, we only considered heterogeneous IP blocks without considering heterogeneous manufacturing technologies. Hence, we cannot fully utilize the potential of heterogeneous integration. In the future, we will study the selection of optimal manufacturing technology and how to design chiplets with hybrid technologies. For instance, one of the most popular topics of computer architecture in recent years is processing in memory (PIM). Many researchers proposed to merge computation logic into memory devices (e.g., DRAM) to minimize data movements and break the communication bottleneck, as shown in Figure 8. However, logic and memory devices have different affinities to manufacturing technologies. Also, a chiplet is a perfect match for PIM since most PIM-based systems cannot be accommodated in a single die. PIM-based systems will be an important application of chiplet design automation.

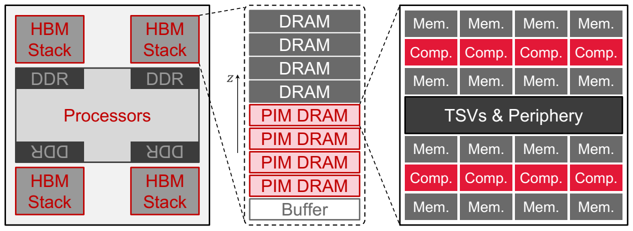


Figure 8: A SiP [27] features both a central processor die and in-memory processing logic, equipped with four DRAM controllers (DDR). Each DDR manages a PIM-HBM stack, consisting of standard DRAM dies, PIM DRAM dies, and a buffer die. The compute logic is placed near the DRAM banks in each PIM DRAM die, enabling parallel access to both DRAM data and computation for improved performance.

5.2 Utilize Partial Functional Chiplets

A chiplet with any defective block will be dropped when modeling the yields. However, the defect of a single block does not necessarily mean the entire chiplet is dysfunctional. It is possible to reuse a broken chiplet with partial functional parts. Core binning is a prevalent manufacturing technique in commercial CPU and GPU products. Vendors test CPU or GPU processors, identify functional parts, and re-brand partial functional dies as new products with less capability. For example, a typical commercial 6-core CPU chip has identical circuits as its' 8-core counterparts, except that there are two defective cores. Prior arts citestow2017:chiplet:design on the architecture design discussed the case with homogeneous cores. For heterogeneous chiplets, the cases are more complicated and need to be explored.

5.3 Interchangeability

We assumed that different types of IP blocks are distinct resources and are not interchangeable. This oversimplification blocks further optimization. For example, a larger SRAM can sometimes replace a smaller one. Our framework cannot capture such a compatibility relationship and will enforce duplication, resulting in higher costs. To make decision-making more holistic, modeling the compatibilities between diverse IP blocks and utilizing such information in design optimization is also very important.

6 CONCLUSION

In this paper, an overview of current progress in chiplet design was presented. Key aspects of chiplet design automation were identified. Preliminary results indicate that automated chiplet design is a promising area, as demonstrated by our algorithm generating two types of chiplets for four SoCs with reduced RE costs, minimal cross-die communication, and significant NRE cost savings.

7 ACKNOWLEDGMENT

This research has been partially supported by a gift to UC Berkeley from Cadence Design Systems and by the National Research

Foundation, Singapore (NRF) under Singapore Hybrid-Integrated Next-Generation μ-Electronics (SHINE) Centre funding program. We thank Size Zheng, Zhihang Yuan, Minghui Chen, Shuhan Liu, Yihan Pang, and Prof. Sanjit Seshia for their help and advice.

REFERENCES

[1] Paul Alcorn. 2023. AMD Instinct MI300 Data Center APU Pictured Up Close: 13 Chiplets, 146 Billion Transistors. Retrieved January 05, 2023 from https://www.tomshardware.com/news/amd-instinct-mi300-data-center-apu-pictured-up-close-15-chiplets-146-billion-transistors

[2] H. B. Bakoglu, G. F. Grohoski, and R. K. Montoye. 1990. The IBM RISC System/6000 processor: Hardware overview. *IBM Journal of Research and Development* 34, 1 (1990), 12–22.

[3] L. Benini and G. De Micheli. 2002. Networks on chips: a new SoC paradigm. *Computer* 35, 1 (2002), 70–78.

[4] Idan Burstein. 2021. Nvidia Data Center Processing Unit (DPU) Architecture. In *2021 IEEE Hot Chips 33 Symposium (HCS)*. 1–20.

[5] Bill Chang, Rajiv Kurian, Doug Williams, and Eric Quinnell. 2022. DOJO: Super-Compute System Scaling for ML Training. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. 1–45.

[6] Jonathan Chang et al. 2007. The 65-nm 16-MB Shared On-Die L3 Cache for the Dual-Core Intel Xeon Processor 7100 Series. *IEEE Journal of Solid-State Circuits* 42, 4 (2007), 846–852.

[7] Chun-Han Chiang, Fu-Yu Chuang, and Yao-Wen Chang. 2020. Unified Redistribution Layer Routing for 2.5D IC Packages. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 331–337.

[8] Pat Conway and Bill Hughes. 2007. The AMD Opteron Northbridge Architecture. *IEEE Micro* 27, 2 (2007), 10–21.

[9] J.A. Cunningham. 1990. The use and evaluation of yield models in integrated circuit manufacturing. *IEEE Transactions on Semiconductor Manufacturing* 3, 2 (1990), 60–71.

[10] Debendra Das Sharma et al. 2022. Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations With Chiplets at Package Level. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 12, 9 (2022), 1423–1431.

[11] Jaideep Dastidar et al. 2022. AMD 400G Adaptive SmartNIC SoC: Technology preview. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. 1–31.

[12] G. Dudeck and J. Dudeck. 1998. Design considerations and packaging of a Pentium(R) Pro Processor based multi-chip module for high performance workstation and servers. In *Proceedings. 1998 IEEE Symposium on IC/Package Design Integration (Cat. No.98CB36211)*. 9–15.

[13] IEEE electronics Packaging Society. 2022. Heterogeneous Integration Roadmap. Retrieved October 15, 2022 from https://eps.ieee.org/technology/heterogeneous-integration-roadmap.html

[14] Denis Foley and John Danskin. 2017. Ultra-Performance Pascal GPU and NVLink Interconnect. *IEEE Micro* 37, 2 (2017), 7–17.

[15] Wilfred Gomes, Slade Morgan, Boyd Phelps, Tim Wilson, and Erik Hallnor. 2022. Meteor Lake and Arrow Lake Intel Next-Gen 3D Client Architecture Platform with Foveros. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. 1–40.

[16] Linley Gwennap. 2021. Chiplets Gain Rapid Adoption: Why Big Chips Are Getting Small.

[17] Per Hammarlund et al. 2014. Haswell: The Fourth-Generation Intel Core Processor. *IEEE Micro* 34, 2 (2014), 6–20.

[18] John L. Hennessy and David A. Patterson. 2019. A New Golden Age for Computer Architecture. *Commun. ACM* 62, 2 (jan 2019), 48–60.

[19] Yuan-Kai Ho and Yao-Wen Chang. 2013. Multiple chip planning for chip-interposer codesign. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[20] Subramanian S. Iyer. 2016. Heterogeneous Integration for Performance and Scaling. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 6, 7 (2016), 973–982.

[21] Iris Hui-Ru Jiang, Yao-Wen Chang, Jiun-Lang Huang, and Charlie Chung-Ping Chen. 2022. Intelligent Design Automation for Heterogeneous Integration. In *Proceedings of the 2022 International Symposium on Physical Design* (Virtual Event, Canada) *(ISPD '22)*. Association for Computing Machinery, New York, NY, USA, 105–106.

[22] Iris Hui-Ru Jiang, Yao-Wen Chang, Jiun-Lang Huang, and Chung-Ping Chen. 2020. Intelligent Design Automation for 2.5/3D Heterogeneous SoC Integration. In *Proceedings of the 39th International Conference on Computer-Aided Design* (Virtual Event, USA) *(ICCAD '20)*. Association for Computing Machinery, New York, NY, USA, Article 125, 7 pages.

[23] Iris Hui-Ru Jiang, Yao-Wen Chang, Jiun-Lang Huang, and Chung-Ping Chen. 2021. Opportunities for 2.5/3D Heterogeneous SoC Integration. In *2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. 1–1.

[24] Norman P. Jouppi et al. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture* (Toronto, ON, Canada) *(ISCA '17)*. Association for Computing Machinery, New York, NY, USA, 1–12.

[25] Moongon Jung, David Z. Pan, and Sung Kyu Lim. 2012. Chip/package co-analysis of thermo-mechanical stress and reliability in TSV-based 3D ICs. In *DAC Design Automation Conference 2012*. 317–326.

[26] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H. Loh. 2016. Exploiting Interposer Technologies to Disintegrate and Reintegrate Multicore Processors. *IEEE Micro* 36, 3 (2016), 84–93.

[27] Sukhan Lee et al. 2021. Hardware Architecture and Software Stack for PIM Based on Commercial DRAM Technology : Industrial Product. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. 43–56.

[28] Sean Lie. 2022. Cerebras Architecture Deep Dive: First Look Inside the HW/SW Co-Design for Deep Learning : Cerebras Systems. In *2022 IEEE Hot Chips 34 Symposium (HCS)*. 1–34.

[29] Wen-Hao Liu, Min-Sheng Chang, and Ting-Chi Wang. 2014. Floorplanning and signal assignment for silicon interposer-based 3D ICs. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.

[30] Lily Looi and Stephan Jourdan. 2009. Transitioning the Intel® next generation microarchitectures (nehalem and westmere) into the mainstream. In *2009 IEEE Hot Chips 21 Symposium (HCS)*. 1–18.

[31] Joe Macri. 2015. AMD's next generation GPU and high bandwidth memory architecture: FURY. In *2015 IEEE Hot Chips 27 Symposium (HCS)*. 1–26.

[32] R. E. Matick and S. E. Schuster. 2005. Logic-based eDRAM: Origins and rationale for use. *IBM Journal of Research and Development* 49, 1 (2005), 145–165.

[33] Samuel Naffziger 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 (ISCA)*. 57–70.

[34] Sergii Osmolovskyi, Johann Knechtel, Igor L. Markov, and Jens Lienig. 2018. Optimal die placement for interposer-based 3D ICs. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. 513–520.

[35] Sivakumar Radhakrishnan, Sundaram Chinthamani, and Kai Cheng. 2007. The Blackford Northbridge Chipset for the Intel 5000. *IEEE Micro* 27, 2 (2007), 22–33.

[36] Alberto Sangiovanni-Vincentelli. 2007. Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design. *Proc. IEEE* 95, 3 (2007), 467–506.

[37] Daniel A. Spielman. 2007. Spectral Graph Theory and its Applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. 29–38.

[38] F. Steenhof, H. Duque, B. Nilsson, K. Goossens, and R.P. Llopis. 2006. Networks on Chips for High-End Consumer-Electronics TV System Architectures. In *Proceedings of the Design Automation & Test in Europe Conference*, Vol. 2. 1–6.

[39] Lisa T. Su, Samuel Naffziger, and Mark Papermaster. 2017. Multi-chip technologies to unleash computing performance gains over the next decade. In *2017 IEEE International Electron Devices Meeting (IEDM)*. 1.1.1–1.1.8.

[40] Sehat Sutardja. 2015. 1.2 The future of IC design innovation. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*. 1–6.

[41] Cadence Design Systems. 2021. Integrity 3D-IC Platform. Retrieved January 05, 2023 from https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/integrity-3dic-platform.html

[42] EE Times. 1999. System-on-chip NRE costs could reach $1 million, says Semico study. Retrieved January 05, 2023 from https://www.eetimes.com/system-on-chip-nre-costs-could-reach-1-million-says-semico-study/

[43] James Warnock et al. 2015. 4.1 22nm Next-generation IBM System z microprocessor. In *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*. 1–3.

[44] John Wuu, Rahul Agarwal, Michael Ciraula, Carl Dietz, Brett Johnson, Dave Johnson, Russell Schreiber, Raja Swaminathan, Will Walker, and Samuel Naffziger. 2022. 3D V-Cache: the Implementation of a Hybrid-Bonded 64MB Stacked Cache for a 7nm x86-64 CPU. In *2022 IEEE International Solid- State Circuits Conference (ISSCC)*, Vol. 65. 428–429.

[45] Jing Xia, Chuanning Cheng, Xiping Zhou, Yuxing Hu, and Peter Chun. 2021. Kunpeng 920: The First 7-nm Chiplet-Based 64-Core ARM SoC for Cloud Services. *IEEE Micro* 41, 5 (2021), 67–75.

[46] Greg Yeric. 2015. Moore's law at 50: Are we planning for retirement?. In *2015 IEEE International Electron Devices Meeting (IEDM)*. 1.1.1–1.1.8.

[47] Doug C. H. Yu. 2014. New System-in-Package (SiP) Integration technologies. In *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*. 1–6.

[48] Zhen Zhuang, Bei Yu, Kai-Yuan Chao, and Tsung-Yi Ho. 2022. Multi-Package Co-Design for Chiplet Integration. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design* (San Diego, California) *(ICCAD '22)*. Association for Computing Machinery, New York, NY, USA, Article 114, 9 pages.