Adaptive Redistribution Layer Routing for Chiplet-Package Co-Design in 2.5D System

ZHEN ZHUANG, The Chinese University of Hong Kong, Hong Kong, Hong Kong
WEISHIUN HUNG, NVIDIA Corp, Taipei, Taiwan
MD ARAFAT KABIR, MangoBoost, Inc, Seattle, United States
YARUI PENG, University of Arkansas, Fayetteville, United States
TSUNG-YI HO, The Chinese University of Hong Kong, Hong Kong, Hong Kong

2.5D packaging has become a popular alternative to integrate advanced logic and memory chiplets for high-performance computing and artificial intelligence systems. In the conventional design flow, chiplets and packages are independently designed and then integrated at the assembly stage. To bridge the gap between chiplet designs and package designs, existing chiplet-package co-design methods iteratively optimize chiplet layouts to improve the performance of the entire system. However, Redistribution Layer (RDL) routing, which finishes the interconnections between chiplets at the package level and significantly affects the system performance, is neglected in the existing co-design flows. Therefore, this paper proposes an effective chiplet-package co-design flow focusing on the RDL routing to optimize the package system performance dynamically. The proposed co-design flow can fill in the missing link, package-level co-optimization, of previous design flows. In the proposed co-design flow, we propose an efficient RDL routing algorithm to iteratively optimize the substrate layout based on the cross-boundary timing context extracted from both chiplets and the package. The proposed RDL routing algorithm has two critical techniques, including 1) a Maximal Independent Set-based (MIS-based) pin assignment method to dynamically optimize the pin positions of nets and 2) a network-flow-based router to generate routing layouts. Experimental results show that the proposed design flow can gradually improve the maximum frequency of a real design to the target performance, 400 MHz.

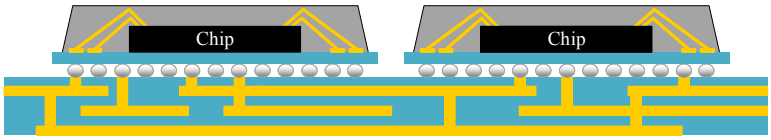
Additional Key Words and Phrases: 2.5D packaging, chiplet-package co-design, redistribution layer routing, timing optimization

1 Introduction

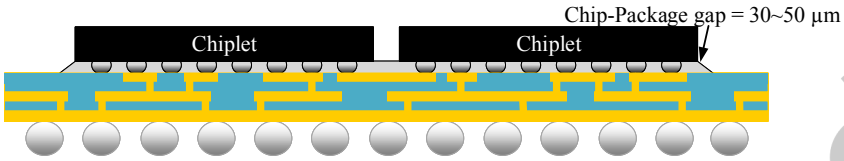
As technologies in 2.5D/3D heterogeneous integration advance, it revealed a glimmer of hope to fulfill the tremendous requirement from the explosive growth of information technologies, such as High-Performance Computing (HPC) and Artificial Intelligence (AI), after the Moore's Law plateauing [1, 2]. Based on advanced chiplet integration techniques [3–6], high-performance chiplet systems can be built to achieve better Power, Performance, Area, Cost, and Time-to-market (PPACT). MediaTek has manufactured many kinds of products, such as high-speed networking chips, by integrating chiplets in a Legoland [7]. Intel has announced their HPC chip with eight cache chiplets on top of a base chiplet [8]. AMD has successfully applied the 3D V-Cache technique for their multiple products [9]. 2.5D integration can scale from Printed Circuit Board (PCB) shown in Figure 1(a) to smaller flip-chip wafer-level packaging shown in Figure 1(b). Both techniques have been extensively developed, but they still face significant challenges in chiplet packaging primarily due to the limitations of the conventional

Authors' Contact Information: Zhen Zhuang, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: zhuangzhen1995@gmail.com; Weishiun Hung, NVIDIA Corp, Taipei, Taiwan; e-mail: wallen11143@gmail.com; MD Arafat Kabir, MangoBoost, Inc, Seattle, United States; e-mail: makabir@uark.edu; Yarui Peng, University of Arkansas, Fayetteville, Arkansas, United States; e-mail: yrpeng@uark.edu; Tsung-Yi Ho, The Chinese University of Hong Kong, Hong Kong, Hong Kong; e-mail: tyho@cse.cuhk.edu.hk.

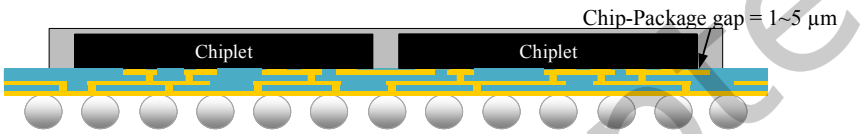
ACM 1557-7309/2025/3-ART
https://doi.org/10.1145/3723043



(a) PCB-based system with ~250 μm pitch.



(b) Flip-chip integration with ~20 μm pitch.



(c) High-density integration with ~1 μm pitch.

Fig. 1. Schematic structures of 2.5D integration of (a) PCB-based system, (b) flip-chip with an organic interposer, and (c) high-density integration such as integrated FOWLP.

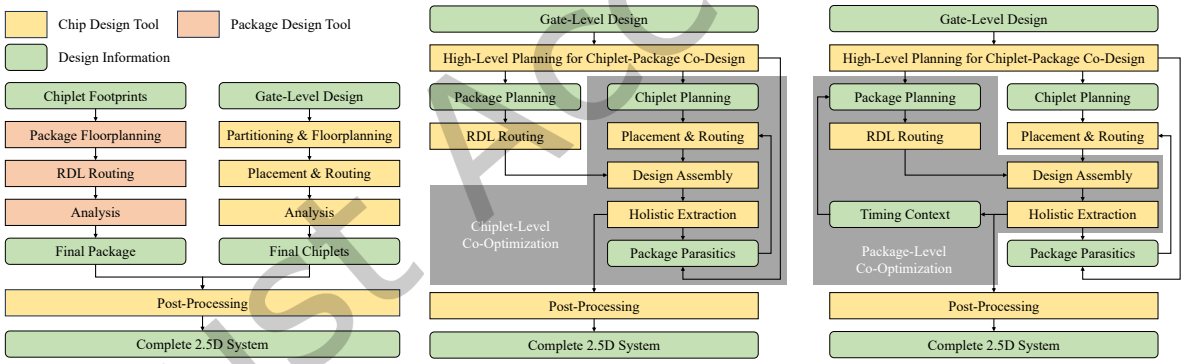


Fig. 2. The comparison between different design flows. (a) Chiplets and packages are independently designed in the traditional design flow. (b) The co-design flow proposed in [10] focuses on optimizing chiplets. (c) The chiplet-package co-design flow proposed in this paper achieves package-level co-optimization.

bump bonding method. To meet the increasing demand for high-bandwidth communication, solder bump shrinks to bumpless connection to fit the needs of high-density integration as shown in Figure 1(c). As the gap scaled to the sub-micron level, new challenges emerged. Existing Electronic Design Automation (EDA) tools separately design chiplets and the package as shown in Figure 2(a). It is handy and intuitive to finish a 2.5D system based on the independent designs of chiplets and packages. However, the independent design flow cannot guarantee the best performance for the integrated system. Therefore, it raised interest in developing co-design flows to conquer the challenges.

Previous work built sophisticated design flows to solve co-design problems focusing on different steps. Huang *et al.* [11] proposed a co-design flow to optimize the Power Delivery Network (PDN) and micro-bump assignment of flip-chip packaging by the k-means clustering algorithm. However, this work does not take into account significant signal routing and chiplets are designed without considering package-level information in the entire design flow. Kim *et al.* [12] performed a detailed design yet its chiplets and package are designed separately without considering their mutual impacts. Both the two conventional design flows independently designing chiplets and packages are shown in Figure 2(a). Kabir *et al.* [10] proposed a design flow for package-chiplet co-design with an optimization loop of system performance by re-configuring the signal assignment as shown in Figure 2(b). However, this work only focuses on the chiplet-level co-optimization. The Redistribution Layer (RDL) routing is not optimized in the co-design flow, resulting in performance degradation at the package level. Specifically, the package layout cannot be adaptively optimized. Furthermore, the critical and challenging pin assignment problem in 2.5D design is not addressed. In this work, we propose a new chiplet-package co-design flow that builds upon the chiplet-level co-optimization approach proposed in [10]. Additionally, a novel sub-flow is embedded to specifically address package-level co-optimization as shown in Figure 2(c).

Concurrent routing methods have been proven they achieve high-quality results compared with sequential routing methods since all the nets are simultaneously routed to achieve global optimization [13–15]. RDL routing problems can be divided into three types: free-assignment routing, pre-assignment routing, and unified-assignment routing. For free-assignment routing, signal nets can freely select terminals from a given pad set. For pre-assignment routing, the terminals of signal nets have been assigned before routing. For unified-assignment routing, signal nets include both free-assignment nets and pre-assignment nets. Fang *et al.* [13] used a network flow formulation for the flip-chip package-board co-design problem. Free and pre-assignment net routing methods are combined to optimize the total wirelength. Lin *et al.* [14] further proposed another network flow formulation for solving unified-assignment RDL routing problems in multi-layer multi-chip designs. Chiang *et al.* [15] formulated the RDL routing problem, provided with pre-assignment nets, into a linear complementarity problem that allows variable widths and spacings. However, they do not consider the optimization of timing issues. Furthermore, none of the above work is estimated by the metrics of final package systems, including performance and power. The nets can be routed octagonally or rectilinearly in advanced packaging [16]. Rectilinear RDL routing is practical. However, octagonal routing increases manufacturing costs and does not scale well into sub-micron pitches due to defects and sharp 45-degree angles This work focuses on rectilinear RDL routing for high-performance applications and the flow is compatible with chip EDA tools that only support Manhattan routing.

At first, we propose a novel chiplet-package co-design flow shown in Figure 2(c) to fill in the missing link based on the previous flow shown in Figure 2(b). Then, we propose a routing methodology based on a network flow algorithm to overcome the drawbacks of the existing algorithms. The proposed algorithm has an optimization scheme that dynamically generates new package routing solutions based on new timing contexts in each loop to find a routing solution of higher quality. The optimized routing solutions can get closer to the performance of a 2D counterpart of the same design. The major contributions are shown below:

- In this paper, we propose a novel chiplet-package co-design flow focusing on the package-level co-optimization to fill in the missing link of previous design flows. The proposed co-design flow can effectively optimize the timing issues to improve the performance of package systems.
- An effective grid-based network flow algorithm is proposed to solve the package-level RDL routing problem. The proposed algorithm has two novel optimization strategies, free layer assignment and via usage control mechanisms. Based on the proposed algorithm, the wirelength of all nets can be concurrently reduced.
- Two methods, a heuristic method and another formulated by the Maximal Independent Set (MIS) algorithm, for pin assignment are introduced. The methods can dynamically update both chiplets and package design

through iterations. By combining the proposed methods with the signal assignment method in [10], the reciprocal adaptation task can be completed for chiplet-package co-design.
- Experimental results show that the proposed design flow can gradually improve the maximum frequency of a real design to the target performance, 400 MHz.

The rest of the paper is organized as follows. Section 2 gives preliminaries of the RDL routing problem. Section 3 details the RDL routing methodology. Section 4 introduces the system architecture and the experimental cases. Section 5 presents the result analysis and Section 6 concludes our contributions and findings.

2 Preliminaries

2.1 Terminologies and Notations

The following terminologies and notations are used:

- $N = \{n_i \mid 1 \leqslant i \leqslant |N|\}$ is the set of nets.
- $C = \{c_i \mid 1 \leqslant i \leqslant |C|\}$ is the set of chiplets.
- $R = \{r_i \mid 1 \leqslant i \leqslant |R|\}$ is the set of redistribution layers.
- $P = \{p_i \mid 1 \leqslant i \leqslant |P|\}$ is the set of pins.

2.2 Design Rule Constraints

The design rules are summarized below:

- Open design rule: Open means any pin of a net is not connected. The routing solution with open nets is regarded as an invalid solution.
- Short design rule: Short is one of the most important metrics. If different metals have overlapped regions, they are regarded as shorted metals.
- Minimum spacing design rule: The minimum spacing between two design components is defined.
- Timing constraint: In this paper, performance targets, such as the maximum frequency, are defined for the entire package system. The co-design results should satisfy the given performance targets.

2.3 Problem Formulation

The package-level chiplet-package co-optimization problem is solved by an iterative RDL routing method to gradually approach the defined performance targets as shown in Figure 3. The RDL routing problem within one iteration is formulated as below:

- **Input:** Netlist, pin arrays, pre-extracted timing contexts, and design rules.
- **Output:** Routing solution without design rule violations.
- **Objective:** Minimize wirelength while maintaining 100 % routability.

3 Methodology

3.1 Methodology Overview

The main purpose of this paper is to iteratively improve chiplet-package timing performance during the RDL routing stage by reducing wirelength. Before each iteration, the timing contexts of chiplets and the package of the last iteration are extracted. Then, the nets are sorted in ascending order based on the pre-extracted timing contexts, i.e., the wire timing budget. Therefore, the chiplet-package timing-critical nets have higher priority to be routed. Finally, the proposed RDL routing algorithm can optimize timing based on the feedback of chiplet-package timing contexts to achieve package-level chiplet-package co-optimization.

Figure 3 shows the workflow of the proposed package-level chiplet-package co-optimization method. The proposed method iteratively optimizes the performance of package systems until STA meets the performance target. For each iteration, the processes are shown below. As sketched in Figure 3, package design starts from pin assignment, which strongly affects the performance upper bound since chiplets are interconnected through the RDL wires and the wirelength dominates the signal delay in the package system. In this work, we formulate the

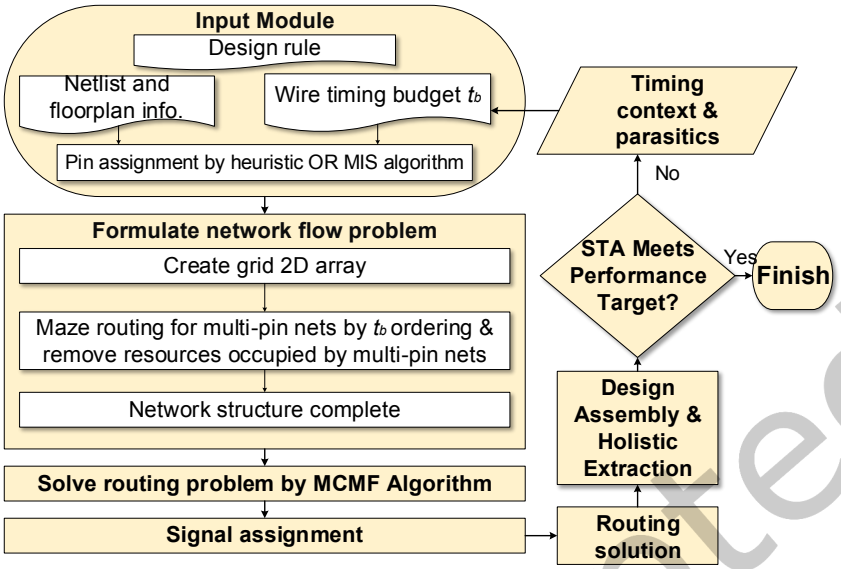


Fig. 3. Workflow of the proposed package-level chiplet-package co-optimization method.

pin assignment problem by two different methods to gradually find the best solution. Then, the network flow formulation is constructed to solve the routing problem. The first step for network flow formulation is to confine the available routing region, transform it into the updated region with desirable routing resources, and construct routing graphs. In the second step, the multi-pin nets are routed by a maze routing algorithm. After removing resources used by multi-pin nets, the updated graph is ready for the Minimum-Cost Maximum-Flow (MCMF) formulation. Next, the routing problem of two-pin nets with MCMF formulation is solved by an MCMF solver. Finally, the signals are assigned to different nets with generated routing layouts.

Integrating pin assignment into the MCMF model makes the network complicated. All the potential I/O pins should be connected to the super sink. If the target design has plenty of I/O pins, solving the MCMF model is very time-consuming. Therefore, we partition the entire design flow into multiple stages, including pin assignment and RDL routing.

The consideration of design rules and the timing constraint is shown below:

- The open design rule is tackled by solving the MCMF model where the output of each net is a pin-to-pin path without open.
- The short design rule is solved based on the defined capacities of the MCMF model. Since the capacity of each edge is set to 1, each edge can only be used by one net. Therefore, different nets cannot have any short violations.
- The minimum spacing design rule is solved by constructing a legal grid for the MCMF model. The size of grids is determined based on the minimum spacing design rule and each grid can only be used by one net, which means the solution cannot violate this constraint.
- The timing constraint is handled by the iterative procedure. The routing solution approaches the performance targets step by step.

In the following content, Section 3.2 will introduce the method to calculate the wire timing budget for timing optimization. Section 3.3 will give the two pin assignment methods. Section 3.4 will provide the details of the proposed MCMF-based routing algorithm. Section 3.5 will present the signal assignment method.

3.2 Wire Timing Budget

If the package and chiplet physical designs are available, they can be assembled in a unified design environment for holistic extraction [10]. The extracted parasitics can be used along with the system-level netlist to perform holistic timing analysis. Then timing contexts can be extracted for each chiplet from the analysis result. However, for the first iteration, there is no physical design to perform parasitic extraction. Thus, we perform timing analysis on the gate-level netlist and extract the timing contexts for each chiplet. For later iterations, timing analysis is performed on the post-layout design after extraction.

The extracted timing contexts of a chiplet include information such as input delay d_i and output delay d_o of each net. For a given package-level net, these values are collected from timing contexts of the chiplets connected by nets. Based on these values, the package wire timing budget t_b of the net is calculated as Equation (1):

$$t_b = c_T - [(c_T - d_o) + (c_T - d_i)] \tag{1}$$

,where c_T stands for the clock period. Based on the calculated timing budget, netlist N is sorted by t_b to make nets with lower t_b get higher priorities of choosing shorter wires to favor the performance.

Although the formulation of t_b is simple, it is good enough to measure the timing sensitivity of different nets since the parameters in the formulation, i.e., d_o and d_i, are extracted from STA. Considering complicated timing metrics about the wirelength and the wire width is very time-consuming. Furthermore, the routing layout varies during the iteration process. The over-detailed analysis may mislead the following optimization iteration. During the RDL routing, the total wirelength of all nets is reduced to optimize chiplet-package latency.

3.3 Pin Assignment Methods

In this work, there is no pre-assigned pin for signal nets in the package-level co-optimization, which means we should solve the free-assignment routing problem. In this way, the entire design flow can achieve global optimization to generate high-quality package designs. Therefore, the pins should be selected from the given pin arrays P in the pin assignment stage before the routing stage. Pin assignment is dynamically updated in each iteration for our reciprocal adaptation scheme, constituting a pivotal strategy to facilitate the real-time exchange of design information between chiplets and the package. Firstly, we propose an efficient heuristic method for pin assignment. However, it cannot handle the nets with the same estimated wirelength. Therefore, we further propose a Maximal Independent Set (MIS)-based method to achieve better package designs.

In each iteration of package-level chiplet-package co-optimization, pin assignment is repeated after holistic extraction and analysis as depicted in `Input Module` in Figure 3. At first, the set of nets N is sorted by the wire timing budget t_b in ascending order. Then, the problem formulation of pin assignment is introduced. Finally, the two methods, the heuristic method and the MIS-based method, are introduced in the following content.

3.3.1 Problem Formulation. The pin assignment problem is formulated as below:

- **Input:** Netlist and pin arrays. Each net has a specified number of pins. The corresponding instance, i.e., chiplet or peripheral I/O, of each pin of a net is specified. The pin arrays include the pin candidates that nets can select.
- **Output:** Selecting the pins in the pin arrays for each net.
- **Objective:** Minimize the total wirelength of all nets. The wirelength is estimated by the Half-Perimeter Wirelength (HPWL) model.

3.3.2 Heuristic Method. This method processes the net in N one by one. For each net, it selects a pin pair/triplet having the smallest estimated wirelength. The estimated wirelength is calculated by the Half-Perimeter Wirelength (HPWL) Model. The proposed heuristic method is very efficient. However, this heuristic method ignores the fact that each net may have multiple choices of pin pairs resulting in the same smallest HPWL. Different choices of one net have significant impacts on the following nets. The simple selection from the pin pairs with the same

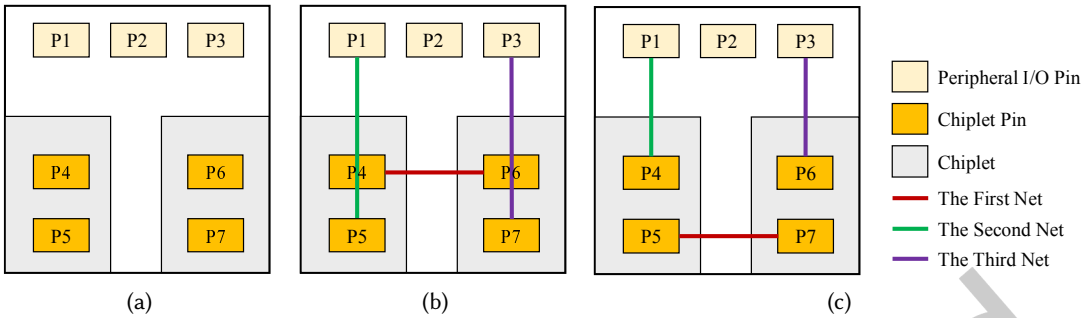


Fig. 4. The illustration of the pin assignment problem. (a) The input of the pin assignment problem. (b) The pin assignment result of the heuristic method. (c) The pin assignment result of the MIS-based method.

HPWL leads to the degradation of the system performance. Therefore, another method is proposed to allow a couple of nets to find their pin pairs simultaneously as introduced below.

3.3.3 MIS-Based Method. A net $n_i \in N$ may have J_i pin pairs (or triplets) leading to the same smallest HPWL. To formulate the pin assignment problem into an MIS problem, we should construct an undirected graph $G_M = (V_M, E_M)$, where G_M represents the graph for problem mapping, V_M represents the vertexes in G_M, and E_M represents the edges in G_M. For the vertex part, each pair/triplet is mapped to a vertex $v_{i,j} \in V_M$, where i and j identify the index of the net and the pin pair/triplet, respectively. For the edge part, we should consider the situations of each net and each pin. For each net n_i, the corresponding pin pairs represented by vertexes are connected to each other to form a clique. In this way, only one pin pair will be selected for n_i by the MIS solver. For each pin p_i, the corresponding pin pairs represented by vertexes are connected to each other to form a clique. In this way, only one pin pair will be selected for p_i by the MIS solver. Based on the proposed graph structure, the optimal pin assignment solutions can be generated without any conflict.

Then, the problem is split into sub-problems described in Equation (2) by dealing with N in one iteration. N_p is the partial netlist satisfying the model mentioned below.

$$N_p = \arg\max_{n_i \in N} \mid N \mid, \tag{2}$$

subject to two constraints. The first constraint is $\mid N_p \mid = \mid MIS(N_p) \mid$ where $MIS(N_p)$ represents the MIS solution of N_p. The second constraint is that N_p is consecutive in N and the net with the smallest t_b must be firstly included in N_p.

When N_p is found with the largest number of nets equaling to the size of solution set $MIS(N_p)$, it is ready for commitment. After committing, pins mapped to vertexes in $MIS(N_p)$ are removed and can no longer be chosen by other unprocessed nets in later sub-problems. N_p is also eliminated from N. We repeat solving this sub-problem, Equation (2), till all nets are solved.

3.3.4 The comparison between the two methods. Figure 4 shows the comparison between the two methods using a simple case. Figure 4(a) illustrates the situation before pin assignment. The pin arrays include peripheral I/O pins and chiplet pins which are highlighted by different color shades. In this case, the pins of three nets should be assigned. Each net has two pins. One net should connect the left chiplet to a peripheral I/O pin. One net should connect the right chiplet to a peripheral I/O pin. One net should connect the two chiplets.

For the heuristic method, the nets are sorted by the wire timing budget t_b in ascending order. The first net is the red net shown in Figure 4(b) which should connect the two chiplets. It will select the pin pair with the minimum HPWL. Four pin pairs, including P1-P4, P3-P6, P4-P6, and P5-P7, have the same HPWL. And the HPWL

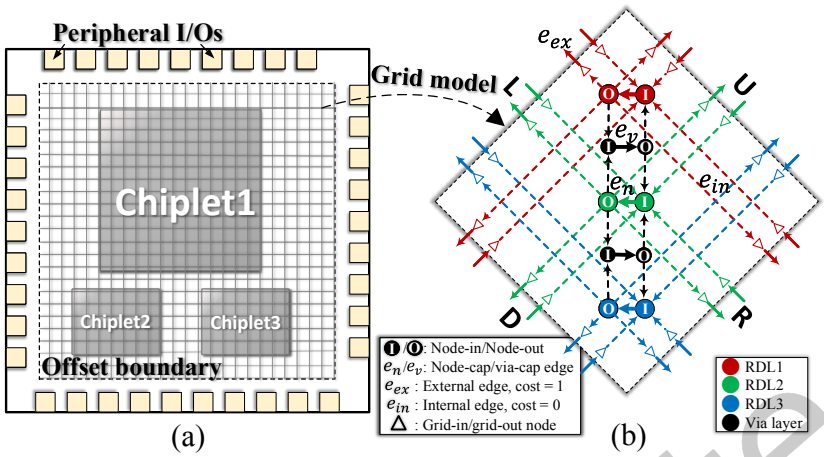


Fig. 5. (a) Floorplan sample with a boundary offsetting (dashed box) inward from peripheral I/Os. (b) Grid model with grid or via nodes in the positions arranged as their physical relative locations. U/D/L/R represents the neighboring grid Up/Down/Left/Right to this grid.

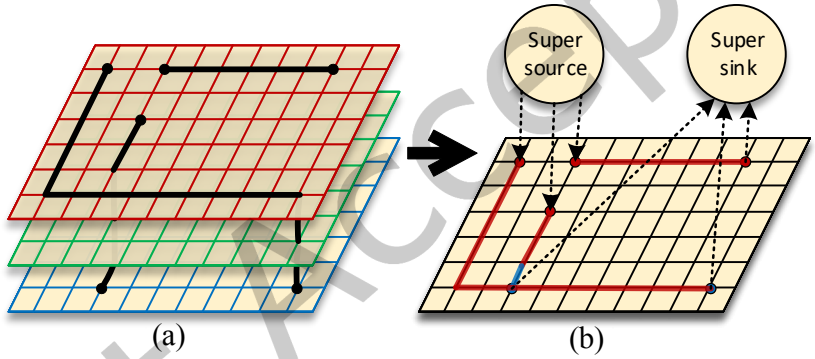


Fig. 6. (a) A routing solution with three nets using RDL1 to RDL3 can be represented as (b) a grid-based network structure.

is the minimum HPWL. Since the heuristic method cannot be aware of the situations of the following nets, the red net may select the pin pair P4-P6 as shown in Figure 4(b). In this way, the other nets can only select pin pair P1-P5 and P3-P7. It is not the optimal solution.

Benefiting from the concurrent optimization ability of the MIS model, the MIS-based method can simultaneously consider the situations of the three nets. Therefore, the MIS-based method can achieve the optimal solution as shown in Figure 4(c).

3.4 MCMF-Based Routing Algorithm

In this work, we formulate the RDL routing problem by a Minimum-Cost Maximum-Flow (MCMF) network flow model. Then, we will introduce the conventional MCMF network flow model. Network flow is a widely used mathematical modeling approach to solve optimization problems. The problem is regarded as generating multiple flows from sources to sinks. The system is represented by a network with nodes and edges, where each edge defines the capacity and cost to limit the edge usage of flows. The objective is to maximize the number of flows and minimize the total cost in the system. Based on the concurrent optimization of the MCMF network flow model, the proposed method can globally optimize routing results.

Table 1. Parameters of our modified 65nm layer stack.

Layer	Purpose	Width	Spacing	Thickness	Epsilon
M1-M7	Chip Internal Routing	TSMC	TSMC	TSMC	TSMC
ILD7	Inter-layer Dielectric	-	-	5 μm	2.0
M8	RDL1	10 μm	10 μm	5 μm	2.2
ILDR1	Inter-layer Dielectric	-	-	5 μm	2.0
M9	RDL2	10 μm	10 μm	5 μm	2.2
ILDR2	Inter-layer Dielectric	-	-	5 μm	2.0
M10	RDL3	10 μm	10 μm	5 μm	2.2
PP	Planar Passivation	-	-	1 μm	4.0
AP	Solder Pads	TSMC	TSMC	TSMC	TSMC

The conventional network flow model cannot be directly used to solve the RDL routing problem in this work. Therefore, we define the specific MCMF network flow model. RDL routing has three main objectives, maximizing the routability, minimizing total wirelength, and limiting the usage of vias. The proposed method can maximize the routability by fulfilling the supplies and demands of flows. The proposed method can minimize total wirelength by minimizing the usage of edges with non-negative costs. The proposed method can limit the usage of vias by setting penalty costs for the edges corresponding to vias. All three objectives can be simultaneously optimized in the proposed model.

The mathematical MCMF network flow model is formally defined as follows:

- Given a directed graph $G = (V, E)$ without parallel edges and a netlist of two-pin nets. $\forall e_i \in E$ has a non-negative cost a_i and capacity 1, where $i \in [1, |E|]$.
- Given source nodes $s_j \in S \subset V$ and the same amount of sink nodes $t_j \in T \subset V$, where $j \in [1, |N|]$. $\forall s_j \in S$ are connected to a super source node s_s and $\forall t_j \in T$ are connected to a super sink node t_s.
- A solution is the flow set satisfying supplies and demands with the minimum cost.

When the layout of the routing region is transformed into the directed graph structure satisfying the above model, the proposed method can generate the flow set. After mapping the flow set to the routing layout, the RDL routing solution is generated. Next, we will introduce the proposed grid partitioning method to transform the routing region into a directed graph structure. Figure 6 shows a mapping example from the routing solution to a grid-based network structure. Based on the grid-based network structure shown in Figure 6(b), the routing solution of three nets can be generated by finding three flows from the source pins connected by a super source to the sink pins connected by a super sink.

To formulate the problem into a network flow structure, we confine the routing region as the offset boundary due to the spacing rule between I/Os and wires as shown in the sample case in Figure 5(a). The bounding box region is partitioned into a set of square grids of the same size. Each grid represents the mid-line of wires or the mid-point of pins. Since the wire width and spacing are 10 μm in all RDLs R as described in Table 1, the grid size is set to 20 μm to satisfy the minimum spacing rule.

The proposed grid model is as illustrated in Figure 5(b) and is composed of the nodes and edges listed below:

- e_n: the edge corresponds to the current grid, named `node-cap edge`. It is a directed edge with a capacity of 1, which means the current grid can only be used by one net. The cost of e_n is 0.
- e_v: the edge corresponds to the current via, named `via-cap edge`. It is a directed edge with a capacity of 1, which means the current via can only be used by one net. The cost of e_v is α. α is a tunable parameter for designers to control via usage.
- n_{in}: the source node corresponds to e_n or e_v, named `node-in`. It belongs to a certain grid or via. A flow must enter by n_{in} to visit a grid or via to ensure that the grid or via can only be visited once.

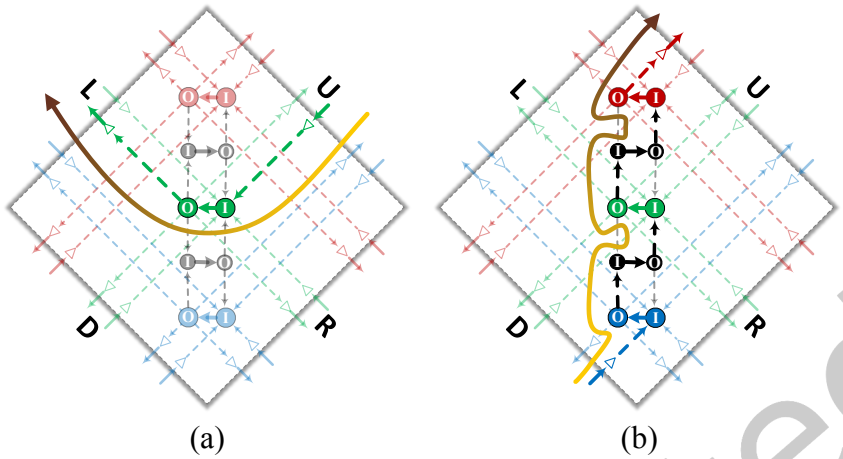


Fig. 7. Routing examples in a grid when (a) making a turn from the grid Up to the grid Left in RDL2 and (b) passing from the grid Down in RDL3 to the grid Up in RDL1.

- n_{out}: the sink node corresponds to e_n or e_v, named `node-out`. It belongs to a certain grid or via. A flow must exit from n_{out} to ensure that the node or via can only be visited once.
- e_{in}: Internal edge directs flow to go in n_{in} or out of n_{out} from or to a neighboring grid. The edge cost is 0.
- e_{ex}: External edge connects to a neighboring grid. Its edge cost is 1 to minimize the total wirelength.

All edges have capacity 1. Note that the grid model is feasible only for two-pin nets. Multi-pin nets are routed firstly by multi-source multi-sink maze routing since they are more likely to get congested than two-pin nets if routed after two-pin nets.

Figure 5(a) illustrates the entire grid graph for routing. Figure 5(b) shows the detailed grid model in the MCMF model corresponding to one grid in Figure 5(a). The red sub-network corresponds to RDL 1. Each boundary of the grid has one edge that enters the grid and one edge that leaves the grid. Signal nets can pass through the grid from any direction in RDL 1 based on this configuration. `Node-cap edge`, `node-in`, and `node-out` are used to make sure that each grid can only be used by one net. Without the capacity limitation of the `node-cap edge`, signal nets can enter the grid from different boundaries and leave the grid from different boundaries, which causes short violations. The black sub-networks correspond to the vias between adjacent metal layers. `Via-cap edge`, `node-in`, and `node-out` are used to make sure that each via can only be used by one net. Without the capacity limitation of `via-cap edge`, a via may be used by two nets. The green sub-network corresponds to RDL 2. The blue sub-network corresponds to RDL 3. They have configurations similar to those of the red sub-network in RDL 1.

Figure 7 shows two grid routing examples by highlighting only the visited nodes and edges. Figure 7(a) shows that a path makes a turn from the grid Up to the grid Left in the second RDL. Based on the proposed grid model, a path can go through the adjacent grids within the same RDL by a `node-cap edge` e_n. Figure 7(a) induces a partial cost of 2. Figure 7(b) shows that a path passes from the grid Down in the third RDL to the grid Up in the first RDL by two vias. Based on the proposed grid model, a path can travel either upward or downward by entering a `via-cap edge` e_v after leaving a `node-out` n_{out}. The path must visit the corresponding `node-in` n_{in} of the `via-cap edge` e_v first. Then, the corresponding via is blocked since the capacity of each `via-cap edge` e_v is only 1. When the path leaves the corresponding `node-out` n_{out} of the `via-cap edge` e_v, it moves to the adjacent RDL by the corresponding via. Figure 7(b) induces a partial cost of $2 + 2 \cdot \alpha$.

This grid model holds the generality for the designs with any number of RDLs. From Figure 5(b), we understand that it can be easily extended to tackle designs with more RDLs simply by expanding the tower-like node array and adding corresponding edges in each grid. The total number of nodes and edges will only increase linearly with the number of RDLs. No matter how many RDLs are included in the model, both total wirelength minimization and via usage control are still simultaneously optimized under this framework.

For each iteration, the routing solution is generated by solving the MCMF model. The difference between the initial solution and the following solutions is how to extract timing contexts from the chiplet-package co-design flow. Since the first iteration does not have the physical design layout, we perform timing analysis on the gate-level netlist and extract the timing contexts for each chiplet.

3.5 Signal Assignment

Like pin assignment, signal assignment is dynamically updated by iterations. It is the final piece of the puzzle to complete the co-design flow. When RDL routing is completed, no pin or wire is assigned with a signal. Each net is assigned to its most suitable wire to break the bottleneck of package overhead on timing performance. After signals are assigned desirably, chiplets can be designed and optimized individually.

3.5.1 Problem Formulation. The signal assignment problem is formulated as below:

- **Input:** Netlist and the routing result including the routing path of nets.
- **Output:** Assigning each net to a routing path based on the timing budget t_b.

As stated in Section 3.2, the initial timing context is estimated from the gate-level netlist without any physical design information. At the end of the first iteration, the physical design of the chiplets and package are available. The extracted parasitic netlist contains all parasitic interactions between the chiplets and package along with the interactions within them. These parasitics are used to perform system-level timing analysis on the design and get the timing contexts of each package net. These contexts are used in the next iteration to re-implement the package design as well as the chiplet designs. For all later iterations, this same methodology is used to get the timing contexts from physical designs. This enables chiplet-package cross-boundary information exchange, the key to our co-design methodology.

After routing and extracting the timing contexts, we hold the wirelength of each routed path and the timing budget t_b of each package net. Nets and wires are paired greedily based on their t_b and the wirelength. Smaller t_b pairs shorter wirelength.

4 Experimental Case Study

To study the impact of our proposed routing strategy in a 2.5D chiplet-package co-design flow, we implemented several design cases by using two different pin assignment methods. We compare them with the designs obtained by the holistic flow proposed in [10], which does not change the package design in its optimization loop as shown in Figure 2(b). We refer to this flow as "Old Flow" and our proposed methodology as "Heuristic" and "MIS" named by their pin assignment methods as shown in Table 2. We use similar settings and design cases of a three-chiplet system presented in [10], which are discussed later in this section. The performance gap means the difference between the max frequency of the 2D design and the max frequency of the chiplet system.

In this work, detailed RDL routing from the package layout combined with all chiplets for holistic analyses, including detailed timing, power, and parasitic extraction. An inter-chiplet static timing analysis (STA) is performed using Synopsys PrimeTime. The clock network is synthesized during the chiplet physical design stage with detailed parasitic information considering all metal layers from package and chiplet layouts. The wire RC parasitics are extracted using Synopsys StarRC and then combined with device delay, which provides detailed timing information for all timing paths, including paths spanning across multiple design domains (chiplets or packages). This is achieved by building a custom PDK with both chiplet and package metal layers, constructing custom DRC/LVS and PEX (parasitic extraction) rules, then merging all layouts from both chiplets and packages

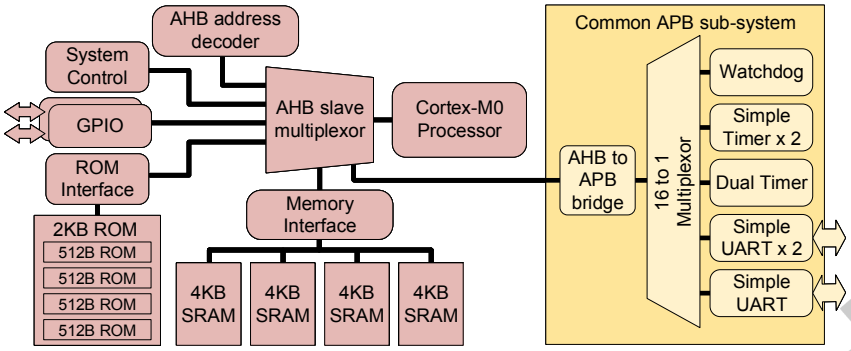


Fig. 8. System architecture of the ARM Cortex-M0-based microcontroller.

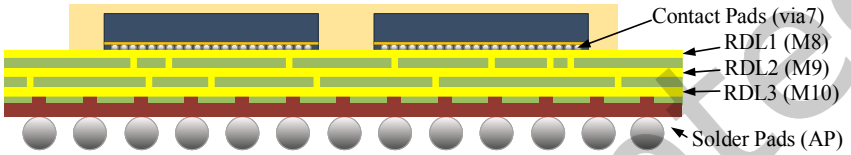


Fig. 9. Package redistribution layer stack of our modified 65nm PDK.

into a single design environment with detailed inter-layer parasitics annotated on complete stitched netlists. Based on the target clock period, the timing slack is then computed for all paths, and the max frequency is computed by the target clock period minus the timing slack of the critical path. Then, the power is calculated using the achievable maximum frequency using STA analysis.

In this work, we manually placed all chiplets, depending on the routing resources available. Since there are few chiplets, the number of routing tracks is computed based on the RDL, pin pitch, and chip edge width. Then, a small spacing gap is allocated to allow routing flexibility and mechanical stability during fabrication.

4.1 System Architecture and Technology Settings

We use an ARM Cortex-M0-based microcontroller system as the experimental design as shown in Figure 8. Like any other microcontroller, the system consists of several peripheral devices like a watchdog timer, two simple timers, a dual-timer, and UART modules. All these devices are part of an APB subsystem. The Cortex-M0 processor core is connected to the rest of the system through an AMBA high-performance bus (AHB). The system has a total of 16KB of memory divided into four banks, 4KB each. For the purpose of 2.5D integration, the system is partitioned into three chiplets. One contains the processor core, all other logic blocks, and 8KB of memory. We call this chiplet the "Core-Chiplet". The rest 8KB of memory is further divided into two memory chiplets, each containing 4KB memory. We refer to them as the "Mem-Chiplets". Due to this partition, all nets in the address bus are three-terminal nets connected to all three chiplets. The data and address bus are two-terminal nets connecting to the `Core-Chiplet` and any one of the two `Mem-Chiplets`.

To compare with the existing work [10], we implement our experimental system in TSMC 65-nm technology. Since a holistic flow requires a unified environment for planning and extraction, we modify the PDK to create a unified chiplet-package technology for our experimental study. Figure 9 shows the package architecture in this work. Table 1 shows the settings of our PDK. The bottom seven metal layers (M1-M7) have the settings specified by TSMC and are used for chiplet internal routing. The top three layers (M8-M10) are modified to mimic the attributes of TSMC configurations. We use 10 μm for both width and spacing on the package layers. The technology stack is characterized by the modified settings to generate the extraction-rule file.

Table 2. Cross-comparison of experimental cases with different methodologies. The total design runtime in Case-3 Final Design is 80, 121, and 42.4 minutes for `Old Flow`, `Heuristic`, and `MIS`, respectively.

Design Case	Case-1	Case-2			Case-3 Initial Iteration		Case-3 Final Design		
Pin Assignment	2D Chip	Old Flow	Heuristic	MIS	Old Flow	Heuristic	Old Flow	Heuristic	MIS
Logic Gates#	24141	20089	20082	20030	20089	20063	20089	20408	20049
Buffer/Inverter#	4760	4714	4737	4806	4822	4877	4670	4924	4805
Total Chip Wirelength (mm)	551.97	510.67	519.76	510.09	512.15	515.76	503.3	511.19	515.75
Package Wirelength (mm)	-	46.83	37.52	32.34	46.83	37.52	46.83	35.88	32.48
Max Frequency (MHz)	400	349	365	397	375	384	387	399	400
Performance Gap	0%	100%	68%	6%	49%	31%	25%	2%	0%
System Power (mW)	20.1	21.8	21.8	21.7	21.8	21.7	21.6	22.2	21.7
Package Design Runtime (s)	-	<1	19	60	<1	18	<1	20	142
Number of Design Iterations	1	1	1	1	1	1	2	3	1

4.2 Design Cases

4.2.1 Case-1 (Reference 2D Design) To have a standard benchmark of performance, we implemented the microcontroller system as a 2D chip. The monolithic chip is implemented using TSMC 65-nm technology using seven metal layers. The gate-level netlist used in this implementation is the same as that of the chiplet system, which is later partitioned into three chiplets for 2.5D implementation. All the standard design steps like PDN, cell placement, clock tree synthesis, routing, etc, are performed using standard chip design tools. The best performance we obtained from this implementation is 400 MHz. The performance of the monolithic chip is used as our target in the other design cases. More details are shown in the Case-1 column of Table 2.

4.2.2 Case-2 (Context-Free 2.5D Designs) This case closely resembles the traditional die-by-die design approach shown in Figure 2(a). In this design case, chiplets are implemented separately without sharing any design or timing context information. However, our routing tool relies heavily on the timing context information. Therefore, we performed static timing analysis (STA) on the gate-level netlist to get the timing contexts to guide the routing tool. Although these timing values are not accurate enough, they can be used as rough estimates. A similar implementation is carried out using the `Old Flow` as shown in Figure 2(b). "Heuristic" and "MIS" represent the proposed chiplet-package co-design flow shown in Figure 2(c) using different pin assignment methods, heuristic method and MIS-based method. Three implementations of Case-2 are presented side-by-side in the Case-2 column of Table 2. All implementations are using the same chiplets. The differences lie only in package designs due to the specific package-level co-optimization and different pin assignment schemes.

4.2.3 Case-3 (Context-Aware Optimized 2.5D Designs) This case employs the iterative co-optimization flow shown in Figure 2(c). In this case, chiplets are assembled with the package for holistic extraction after the physical designs are ready. The extraction result is used to perform STA on the entire system and extract the timing context for each chiplet. These timing contexts are used to re-implement the system layout in the next iteration. In the `Old Flow` shown in Figure 2(b), the timing contexts are used to update the chiplet designs only without updating the package design. In our co-design flow shown in Figure 2(c), the timing contexts are used to update all chiplets as well as the package. In this way, we complete the better chiplet-package co-optimization loop for 2.5D system design. The three implementations took different numbers of iterations to converge. In the right column of Table 2, The final designs are compared side-by-side.

5 Experimental Results

The package design tool was implemented in C++ and tested on a PC with 3.6 GHz CPU and 64 GB memory. We used the LEMON library [17] to implement the MCMF solver and KaMIS [18] to solve our MIS problem. Table 2 summarizes the performance outcomes of the cases described in the previous section. We will discuss the frequencies, design runtime, system power, and wirelength in this section.

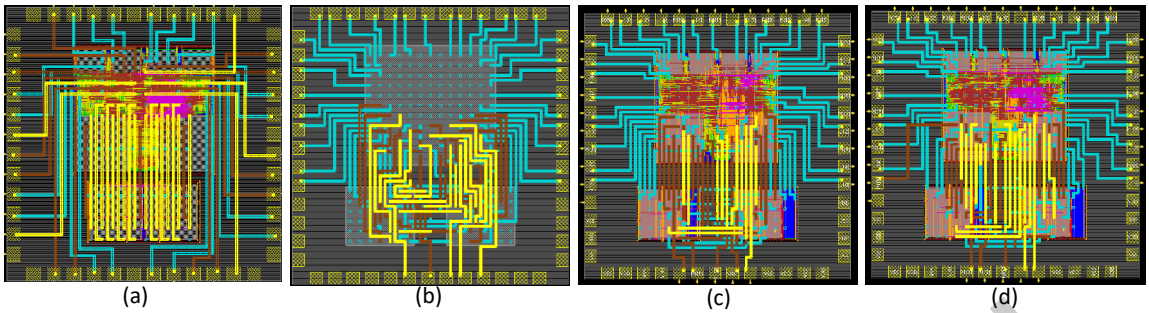


Fig. 10. Design layouts of (a) package final design in Case-3 using the `Old Flow` [10] (b) and the proposed co-design flow with heuristic pin assignment method in Case-3 Initial Iteration, (c) MIS in Case-2, and (d) MIS in Case-3 Final Design.

5.1 Maximum Frequency

5.1.1 Case-1 (Reference 2D Design) The results of the reference design are regarded as the performance targets. The reference design is tuned to achieve the best achievable performance, 400 MHz frequency. All the 2.5D implementations are optimized to achieve the performance as good as that of the 2D design. However, unlike Case-1, the 2.5D implementations have long and wide package wires with significant parasitics [10]. Due to the package-level overhead, the 2.5D systems can hardly run as fast as the 2D implementation. This is evident in the `Max Frequency` row of Table 2.

5.1.2 Case-2 (Context-Free Designs) Case-2 closely resembles the traditional die-by-die approach. Chiplets are designed without taking into account any other part of the system. However, the package routing is performed based on the system design information. In our proposed methodology, we utilize the timing contexts extracted through STA on the synthesized netlist. While using the same methodology, we compare the performance of two different pin assignment methods as introduced in Section 3.3. `Old Flow` achieves a system performance of 349 MHz, which has a performance gap of 51 MHz with respect to the reference 2D design. `Heuristic` shows a maximum frequency of 365 MHz, which is 35 MHz behind Case-1. This performance gap is a reflection of the package overhead due to less careful routing and signal assignment. Moreover, chiplet optimization is performed without the knowledge of the rest of the system and so drivers are not properly adjusted to compensate for the package wireload. Whereas, `MIS` achieves a system performance of 397 MHz, which is a reduction in the performance gap by 94 %, and it is already close enough to the performance target. Note that, no iterative optimizations are performed and the chiplets are designed without any timing context information. Thus, this performance improvement from `Heuristic` and `MIS` is solely coming from different pin assignment strategies.

5.1.3 Case-3 (Context-Aware Optimized 2.5D Designs) The designs are implemented using iterative optimization flows. The final design layout of `Old Flow`, `Heuristic`, and `MIS` are shown in Figure 10. Although `Old Flow` uses the timing contexts extracted from the physical design to improve the chiplet designs, it does not make any effort to improve the package design. Our proposed flow utilizes the timing contexts to optimize the chiplets as well as the package design. As a result, both `Heuristic` and `MIS` beats the designs from `Old Flow` in performance values in all the iterations, which is evident in Table 2.

On the aspect of `Number of Design Iterations`, `Old Flow` achieves 387 MHz in the second iteration. The following iterations do not further improve the system performance. However, `Heuristic` improves the system performance till the third iteration. Because the final performance of `Old Flow` is already bounded by the initial package design. If the initial package design is poor, even the iterative optimization cannot make up for the package overhead. Furthermore, `MIS` takes only one iteration to reach the target performance.

Our proposed package routing methodology can dynamically adapt package design with the changing chiplet designs in the iterative optimization loop by starting from rearranging pin assignment aided by extracted timing

contexts as shown in Figure 2. After the package is routed, the following signal assignment step also makes use of the timing contexts and wirelength of each routed path to update the signals for chiplet design, which means chiplets adapt their design to the provided package design. After chiplets are done, package and chiplets can be assembled for STA. This completes an iteration. We run the iterative loop till performance converges.

`Heuristic` achieves the final performance of 399 MHz, which is almost as good as that of the 2D implementation, even with the additional inter-chiplet overhead. `MIS` further makes the package possible to meet the timing budget for all signals with its carefully designed pin assignment scheme and finally reached the target performance of 400 MHz in one iteration as shown in `Number of Design Iterations`. It is worth mentioning that `MIS` in Case-2 and Case-3 gives quite the same layout patterns (Figure 10(c, d)) owing to that Case-2 already reached very high performance and thus only a little margin is left to Case-3.

In our proposed methodology in Figure 3, the package and chiplets are adapting to each other reciprocally in every iteration. If the chiplet optimization tool fails to meet the target performance because of a few critical signals, the package router compensates for the delay by adjusting the package routing and/or signal re-assignment. Similarly, if the router is not being able to meet the timing budget, the chiplet optimization tool works harder to insert stronger drivers to meet the performance target. Due to these co-optimization efforts, significant performance improvement can be found in `Max Frequency` in `Heuristic` and `MIS` compared to `Old Flow` in Table 2.

Accurate extraction of mutual inductance and coupling capacitance between chiplets and the package is also critical for Signal/Power Integrity (SI/PI) analysis. `MIS` could perform better than `Heuristic` for its more regular wire patterns (Figure 10(b, c)) for signal and power integrity. MIS-based routing eliminates some turning points and short segments, which means it can further improve signal integrity by reducing impedance mismatch, antenna effects, and signal reflection in high-speed transmission lines. With our cross-boundary extraction, this adaptive layout design scheme could also help improve PDN design and mitigate noise and crosstalk. It would be also possible to take SI/PI impact into account in the future co-design flow to achieve high-quality package design.

5.2 Design Runtime

Design runtime is a sum of both chiplet and package design runtime from all iterations. Chiplet design runtime for one iteration is around 40 mins for all cases, which makes the `Package Design Runtime` in Table 2 almost negligible. As previously discussed in Section 5.1.3, despite MIS being the longest in `Package Design Runtime` in Case-3, it still benefits us with much higher productivity and solution quality because the chiplet design runtime dominates the total design time for its least `Number of Design Iterations`. Less iteration, less time consumed for chiplet design tools.

5.3 System Power and Wirelength

The `System Power` in Table 2 is computed at 400 MHz for all designs. It is observable that all designs have almost the same power values in all iterations except the 2D design. In the final iteration of `Heuristic`, it has a little higher power compared to others in the final design. This is due to the increased number of buffers and inverters and other logic gates driving the package wires. This is the price paid to obtain the improved system performance. The final design of `MIS`, however, still maintains moderately better `System Power` than `Heuristic` thanks to its high-quality package design.

In Case-3, the final `Package Wirelength` dramatically reduced from 46.83mm to 32.48mm from `Old Flow` to `MIS`. Furthermore, we can find that `Total Chip Wirelength` increases, which is also the cost of improving the performance. Since routability is guaranteed to be 100%, it is not reported. Otherwise, the MCMF solver will fail to find a solution due to the lack of routing resources.

6 Conclusion and Future Work

In this work, we propose a chiplet-package co-design flow focusing on the package-level co-optimization to fill in the missing link of previous design flows. For package-level co-optimization, we propose an RDL routing methodology with a reciprocal adaptation scheme. It can achieve the simultaneous optimization of signal nets to maximize routability, restrain via usage, minimize total wirelength, and fulfill the timing constraint. Furthermore, the proposed RDL routing methodology integrates two pin assignment methods, including a heuristic method and a Maximal Independent Set (MIS)-based method, which can be dynamically re-configured. Experimental results show that the proposed chiplet-package co-design flow can effectively reach the target performance to 400 MHz.

In the future, we will extend the proposed method to support octagonal routing by post-processing techniques, such as changing corners to octagonal wires. We will extend this work to support the routing problem of diff-pair nets and bus nets. For example, if we bind each diff-pair net group or bus net group as one net, the proposed method can be applied to the routing problem of diff-pair nets and bus nets. We will test the proposed method based on more practical designs.

Acknowledgment

The research work described in this paper was conducted in the JC STEM Lab of Intelligent Design Automation funded by The Hong Kong Jockey Club Charities Trust. This work is jointly supported by the Research Grants Council of Hong Kong SAR (No. CUHK14211324).

References

[1] William Chen and Bill Bottoms. 2019. Heterogeneous integration roadmap: Driving force and enabling technology for systems of the future. In *Proceedings of Symposium on VLSI Technology*. IEEE, T50–T51.

[2] 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 International Symposium on Physical Design*. 105–106.

[3] SY Hou, Chien Hsun Lee, Tsung-Ding Wang, Hao Cheng Hou, and Hsieh-Pin Hu. 2023. Supercarrier redistribution layers to realize ultra large 2.5 D wafer scale packaging by CoWoS. In *Proceedings of IEEE Electronic Components and Technology Conference (ECTC)*. IEEE, 510–514.

[4] Hiroshi Kudo, Masaya Tanaka, Takamasa Takano, Satoru Kuramochi, Kazuyoshi Togashi, and Seiichi Yoshimi. 2023. Signal Integrity of 2-μm-Pitch RDL Interposer for High-Performance Signal Processing in Chiplet-Based System. In *Proceedings of IEEE Electronic Components and Technology Conference (ECTC)*. IEEE, 531–536.

[5] Yu-Chen Hu, Yu-Min Liang, Hsieh-Pin Hu, Chia-Yen Tan, Chih-Ta Shen, Chien-Hsun Lee, and SY Hou. 2023. CoWoS architecture evolution for next generation HPC on 2.5 D system in package. In *Proceedings of IEEE Electronic Components and Technology Conference (ECTC)*. IEEE, 1022–1026.

[6] Lihong Cao, Chen-Chao Wang, Chih-Yi Huang, and Hung-Chun Kou. 2023. Advanced Packaging Design Platform for Chiplets and Heterogeneous Integration. In *Proceedings of IEEE Electronic Components and Technology Conference (ECTC)*. IEEE, 1032–1037.

[7] Chih-Ming Hung. 2023. Semiconductor Chip Design in a Legoland. In *Proceedings of IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 1–4.

[8] Wilfred Gomes, Altug Koker, Pat Stover, Doug Ingerly, Scott Siers, Srikrishnan Venkataraman, Chris Pelto, Tejas Shah, Amreesh Rao, Frank O'Mahony, et al. 2022. Ponte Vecchio: A multi-tile 3D stacked processor for exascale computing. In *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 42–44.

[9] 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 *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC)*, Vol. 65. IEEE, 428–429.

[10] MD Arafat Kabir and Yarui Peng. 2021. Holistic Chiplet–Package Co-Optimization for Agile Custom 2.5-D Design. *IEEE Transactions on Components, Packaging and Manufacturing Technology* 11, 5 (2021), 715–726.

[11] Ming-Yu Huang, Hung-Ming Chen, Kuan-Neng Chen, Shih-Hsien Wu, Yu-Min Lee, and An-Yu Su. 2020. A Design Flow for Micro Bump and Stripe Planning on Modern Chip-Package Co-Design. In *Proceedings of IEEE Electronic Components and Technology Conference (ECTC)*. IEEE, 2236–2241.

[12] Jinwoo Kim, Gauthaman Murali, Heechun Park, Eric Qin, Hyoukjun Kwon, Venkata Chaitanya Krishna Chekuri, Nael Mizanur Rahman, Nihar Dasari, Arvind Singh, Minah Lee, et al. 2020. Architecture, chip, and package codesign flow for interposer-based 2.5-D chiplet integration enabling heterogeneous IP reuse. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28, 11 (2020), 2424–2437.

[13] Jia-Wei Fang, Martin DF Wong, and Yao-Wen Chang. 2009. Flip-chip routing with unified area-I/O pad assignments for package-board co-design. In *Proceedings of Design Automation Conference*. 336–339.
[14] Bo-Qiao Lin, Ting-Chou Lin, and Yao-Wen Chang. 2016. Redistribution layer routing for integrated fan-out wafer-level chip-scale packages. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 1–8.
[15] Chun-Han Chiang, Fu-Yu Chuang, and Yao-Wen Chang. 2020. Unified redistribution layer routing for 2.5 D IC packages. In *Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 331–337.
[16] Pruek Vanna-Iampikul, Lingjun Zhu, Serhat Erdogan, Mohanalingam Kathaperumal, Ravi Agarwal, Ram Gupta, Kevin Rinebold, and Sung Kyu Lim. 2023. Glass Interposer Integration of Logic and Memory Chiplets: PPA and Power/Signal Integrity Benefits. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.
[17] Balázs Dezső, Alpár Jüttner, and Péter Kovács. 2011. LEMON–an open source C++ graph template library. *Electronic notes in theoretical computer science* 264, 5 (2011), 23–45.
[18] KaMIS. https://karlsruhemis.github.io/. ([n. d.]).

Received 29 August 2024; revised 26 January 2025; accepted 27 February 2025