Hierarchical Partitioning-Based Interchip Redistribution Layer Routing for Fan-Out Wafer-Level Packaging

Haoyang Xu, Xing Huang, *Senior Member, IEEE*, Zhen Zhuang, Zhiwen Yu *Senior Member, IEEE*, Bin Guo, *Senior Member, IEEE*, Kai-Yuan Chao, Bei Yu, *Senior Member, IEEE*, Tsung-Yi Ho, *Fellow, IEEE*, and Martin D. F. Wong, *Life Fellow, IEEE*

***Abstract*—In modern advanced fan-out wafer-level packaging (FOWLP), redistribution layers (RDLs) are used to implement efficient interconnections among chips, where both flexible vias and irregular pad structures can be deployed to realize highly integrated heterogeneous systems. With the sharp increase in chip densities, however, it becomes challenging to fully utilize the routing resources in RDLs. The routing graph without optimized resource planning can lead to significant deterioration in the solution quality of RDL design. Furthermore, prior work determines via locations in the RDLs before identifying wire routes, thus offsetting the advantages of flexible vias and irregular pads. To systematically solve the RDL routing problem in FOWLP, in this article, we propose a hierarchical partitioning-based interchip routing method considering both flexible vias and irregular pads, so that optimized RDL solutions can be generated automatically and efficiently. The proposed method includes the following key techniques: 1) a channel planning method for optimized routing region partition and a dynamic routing graph-based pathfinding method for efficient global routing; 2) a binary-search-based hierarchical tile segmentation method for conflict removal and wirelength reduction; 3) a novel monotonicity-analysis-based access point optimization method and a staggered via planning strategy for both wirelength and detour reduction; and 4) a geometry-based detailed routing algorithm for innertile wire routing. The experimental results demonstrate that the proposed algorithm leads to 100% routablility and 4.8% wirelength reduction on average compared to the state-of-the-art work, with a maximum reduction rate of 10.3% across all the benchmarks.**

***Index Terms*—Fan-out wafer-level packaging (FOWLP), flexible vias, hierarchical partitioning, irregular pads, redistribution layer routing.**

Received 20 October 2024; revised 13 February 2025; accepted 31 March 2025. Date of publication 4 April 2025; date of current version 22 October 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62572396; in part by the National Key Research and Development Program of China under Grant 2024YFB4505502; in part by the National Foreign Experts Program under Grant H20240911; in part by the National Natural Science Fund of China for Excellent Young Scientists Fund Program (Overseas); in part by the National Science Fund for Distinguished Young Scholars under Grant 62025205; and in part by the Natural Science Foundation of Fujian Province under Grant 2024J01984. This article was recommended by Associate Editor Y. Lin. *(Corresponding author: Xing Huang.)*

Haoyang Xu, Xing Huang, and Bin Guo are with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: xuhy@mail.nwpu.edu.cn; xing.huang@nwpu.edu.cn; guob@nwpu.edu.cn).

Zhen Zhuang, Bei Yu, and Tsung-Yi Ho are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: zhuangzhen1995@gmail.com; byu@cse.cuhk.edu.hk; tyho@cse.cuhk.edu.hk).

Zhiwen Yu is with the College of Computer Science and Technology, Harbin Engineering University, Harbin 150009, China, and also with the School of Computer Science, Northwestern Polytechnical University, Xi'an 710129, China (e-mail: zhiwenyu@hrbeu.edu.cn).

Kai-Yuan Chao is with Siemens EDA, Siemens Digital Industries Software, Wilsonville, OR 970970 USA (e-mail: kevin.chao@siemens.com).

Martin D. F. Wong is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong (e-mail: mdfwong@hkbu.edu.hk).

Digital Object Identifier 10.1109/TCAD.2025.3558145

I. Introduction

ADVANCED packaging technologies have shown a growing trend toward devices with high I/O density, ultrathin profiles, and high reliability. As a new paradigm of heterogeneous multichip system integration, the fan-out wafer-level packaging (FOWLP) technology has been proven to be very effective in I/O density enhancement [1], power efficiency improvement [2], [3], and thermal performance improvement [4] for various advanced products, e.g., Samsung smartphone series [5] and Apple iPad series [6], thus overcoming the bottlenecks caused by limited chip footprint. In particular, the ultrahigh-density FOWLP technology of the Taiwan Semiconductor Manufacturing Company (TSMC) has demonstrated significant advantages over existing packaging technologies [7], [8], [9], [10], [11]. In such an integrated fan-out package, some additional mental layers, also referred to as redistribution layers (RDLs), are utilized to establish efficient interconnections among heterogeneous chips [12], leading to highly integrated processing systems. Up to now, TSMC's FOWLP technology has been successfully applied in many high-performance computing systems, such as artificial intelligence accelerators and cloud computing devices [13], [14], [15].

Fig. 1 shows the schematic of a FOWLP structure, where the molding compound that covers multiple chips is placed at the top side of the entire package to provide physical protection. Beneath these chips, several RDLs are deployed to redistribute signal connections. These RDLs consist of via layers and metal layers stacking alternately, where via layers are used to construct interlayer connections for signal transmission between adjacent layers while wires in the metal layers provide intralayer connections. At the bottom side of the package, RDLs are further connected to the package substrate through bump pads, thus forming a complete package structure. With the FOWLP structure above, wire connections between I/O pads of different chips, also referred to as interchip interconnections, can be implemented efficiently through both interlayer and intralayer routing in the RDLs [16], [17].

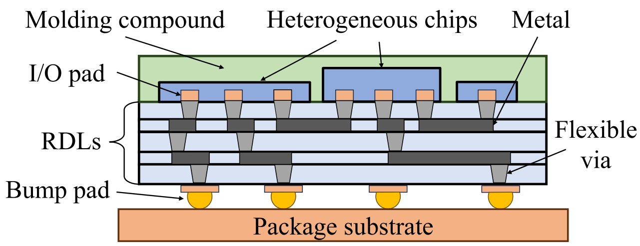

Fig. 1. Schematic of a FOWLP structure.[1]

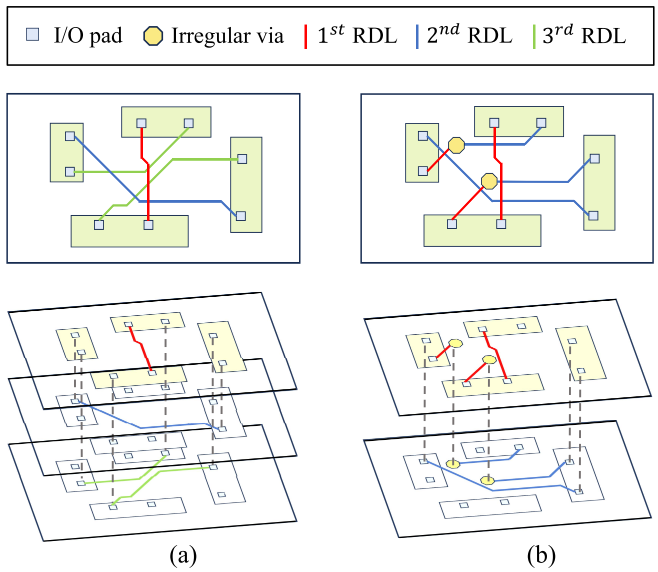

Fig. 2. Traditional packaging technology versus advanced FOWLP technology regarding the RDL routing design. (a) RDL routing solution of four chips using traditional packaging technology and (b) RDL routing solution using the FOWLP technology.

To ensure the overall performance of the integrated heterogeneous multichip system, the aforementioned interchip RDL routing needs to be dealt with carefully, so that optimized via-and-wire networks can be constructed systematically as per the predefined routing nets. On the other hand, traditional packaging technologies allow only fixed vias and regular pads in a package [18], where I/O and bump pads with the same size and pitch are deployed as regular 2-D arrays. These pads are aligned with fixed vias punching through RDLs for interlayer interconnections and this, undoubtedly, reduces the flexibility of RDL routing while the solution space is heavily restricted. For example, since each routing net can only be implemented within a single layer when fixed vias are employed, layer assignment has to be performed before routing so that potential wire crossings can be avoided. In contrast, flexible vias that can be placed arbitrarily in the RDLs and irregular pads with different sizes/pitches have already been supported by the FOWLP technology [11], [19], enabling efficient heterogeneous integration. More importantly, due to the vast flexibility of RDL routing in FOWLP, the overall package cost can be further reduced. For example, since

[1]We note there are many FOWLP derivatives in this fast-moving 3-D-IC industry. Without loss of generality, this basic model does not cover extra via types like TIV (through InFO via) or even more complicated CoWoS-L where embedded silicon bridge small interconnect chips are buried between RDL routing layers. Moreover, the real package substrate is much bigger and thicker than the cartoon drawing of this figure.

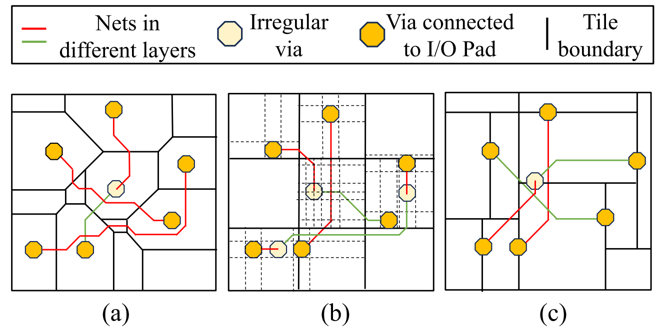

Fig. 3. Comparison among different interchip RDL routing methods with flexible vias. (a) solution generated by [16] (wirelength = 392), (b) solution generated by [17] (wirelength = 384), and (c) optimized solution (wirelength = 339).

only fixed vias and regular pads are employed in Fig. 2(c) and (a), three RDLs are used to route the nets among four chips. In Fig. 2(c) and (b), both the number of RDLs and the total wirelength are reduced due to the adoption of FOWLP technology.

Up to now, several methods have been proposed to address the interchip RDL routing problem. In [20] a minimum cost maximum-flow algorithm and a ring-by-ring routing method are proposed to generate interchip connections. In [21] a directed acyclic graph-based algorithm is proposed to generate routing solutions with a minimized number of RDLs. Both methods, however, are based on fixed vias punching through all RDLs. The vast flexibility of via placement and wire routing has been ignored, resulting in unsatisfactory solutions. In contrast, since both flexible vias and irregular pads need to be considered in FOWLP, this makes the interchip RDL routing problem very complicated. In recent years, machine learning has been increasingly applied to address routing design of integrated circuits. For example, in [22] an asynchronous reinforcement learning framework is proposed to optimize net order in detailed routing. Reference [23] employs deep reinforcement learning to realize custom circuit routing while adhering to industrial design rules. In [24], a generative pretrained transformers (GPT)-based method is proposed for PCB routing, which transforms the routing task into a token prediction problem.

In particular, in [16] a Voronoi diagram-based method is proposed to deal with the placement of flexible vias in the RDL routing. However, since vias in [16] are placed irregularly within tiles before detailed routing, the total wirelength is increased significantly so that vias can be bypassed in the routing stage as illustrated in Fig. 3(d) and (a). In [17] a weighted-MPSC-based layer assignment method and an octagon model-based algorithm are proposed to deal with wire routing and irregular pads, respectively. Furthermore, a linear programming (LP)-based layout optimization method is adopted to realize via arrangement and wirelength reduction. This method, unfortunately, generates many tiny fragment tiles and cannot find the optimum via locations due to the lack of an overall planning of the routing region, leading to some redundant vias and extra wire detours as shown in Fig. 3(d) and (b). In contrast, Fig. 3(d) and (c) shows an optimized routing solution generated by proposed method, where vias are placed at tile boundaries to avoid unnecessary wire detours.

Moreover, by taking intertile and intratile pathfindings into account simultaneously, both wirelength and extra vias are reduced simultaneously.

In view of the importance of interchip RDL routing in FOWLP as well as the drawbacks suffered by existing methods, in this article, we take both flexible vias and irregular pads into account and present an efficient routing flow to solve this problem in a global manner. Our contributions are listed below.

1) We propose a hierarchical partitioning-based method that can fully utilize routing resources in the RDLs, leading to efficient interchip connections with minimized wirelength and 100% routability.
2) We propose a channel planning method for optimized routing region partition. Moreover, a novel dynamic routing graph (DRG)-based pathfinding method is adopted to generate efficient global routing solutions.
3) We propose a binary-search-based tile segmentation method to improve routability while reducing the total wirelength. Moreover, a chord model-based method is employed to generate efficient layer assignment results.
4) We propose a function monotonicity analysis-based method for the placement optimization of access points. Moreover, a staggered via planning strategy is employed to reduce wire detour for both global routing and layer assignment.
5) We propose an efficient detailed routing method that integrates pattern routing, corner routing, and contour routing together. Experimental results confirm that the proposed method achieves better solutions compared with the state-of-the-art techniques.

II. Terminologies, Design Rules, and Problem Formulation

In this section, we first describe the terminologies and design rules used in the RDL routing. Afterward, the interchip RDL routing problem in FOWLP is formulated.

A. Terminologies and Design Rules

In contrast to fixed vias that punch through all the RDLs in traditional packaging technologies, vias can be placed arbitrarily in the RDLs to increase the flexibility of multichip integration and further improve the overall performance of the FOWLP structure [11]. Moreover, I/O pads attached to the top RDL can also be placed irregularly to support heterogeneous integration [25]. In this work, we take both flexible vias and irregular pads into consideration during interchip RDL routing. First, the standard definitions of terminologies used in this article are described as follows.

1) *I/O Pads:* Rectangular pads attached to the top RDL for chip-to-chip and chip-to-package contacts.
2) *Bump Pads:* Octagonal pads attached to the bottom RDL for package-to-board contacts.
3) *Irregular Pads:* I/O or bump pads with different sizes and pitches.
4) *Vias:* Metal pillars used for interlayer connections.
5) *Top Vias:* Vias attached to I/O pads in the top RDL.

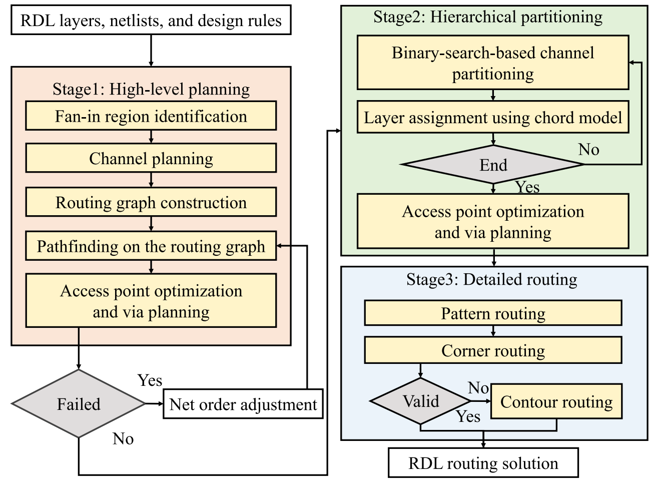

Fig. 4. Overall flow of the proposed algorithm.

6) *Flexible Vias:* Vias that can be placed arbitrarily in the RDLs.
7) *Via Layer:* An RDL used for via deployment.
8) *Wire Layer:* An RDL used for mental wire connections.
9) *Interchip Nets:* Predefined routing nets between chips. Each net is associated with a pair of I/O pads that belong to different chips.

Some notations used in the article are listed below.

1) $P = \{p_i \mid 1 \leq i \leq |P|\}$: A set of I/O pads.
2) $N = \{n_i \mid 1 \leq i \leq |N|\}$: A set of interchip nets.
3) $L = \{l_i \mid 1 \leq i \leq |L|\}$: A set of routing layers.
4) w_e: The width of wires.
5) w_v: The width of vias.[2]
6) δ: The minimum spacing between vias and wires of different nets.

In addition to flexible vias and irregular pads, the design rules that need to be considered in the RDL routing are as follows.

1) *X-Architecture Wires:* Wire segments can only be routed in ±90°, 45°, and 135° directions.
2) *Noncrossing Rule:* Any two interchip nets cannot be implemented across each other in an RDL.
3) *Minimum Spacing Rule:* Any two components need to satisfy the specified minimum spacing, including wires, vias, and I/O pads.
4) *Connection-Angle Rule:* A 90° or 135° turn is allowed if two wire segments are connected. A 45° turn is not allowed for better manufacturability [26], [27].

B. Problem Formulation

Based on the above analysis, the interchip redistribution layer routing problem of heterogeneous multichip system integration in modern FOWLP can be formulated as follows.

Given a set N of interchip routing nets, a set P of I/O Pads (irregular), a set L of RDL layers, and design rules, establish the connections of all the nets with flexible vias such that the routability is maximized, the total wirelength is minimized, and no design rules are violated.

[2]A via is modeled as a regular octagon and w_v is set to the width of its bounding box.

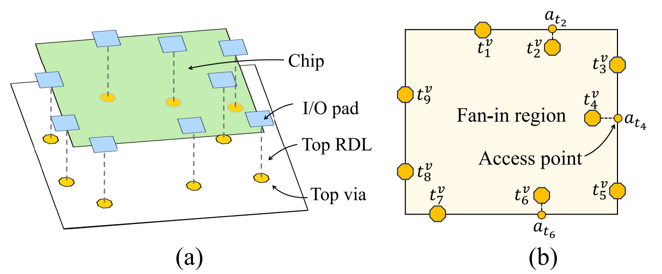

Fig. 5. Illustration of fan-in region identification. (a) Top via insertion for a chip with nine I/O pads and (b) fan-in region and access points of the chip shown in (a).

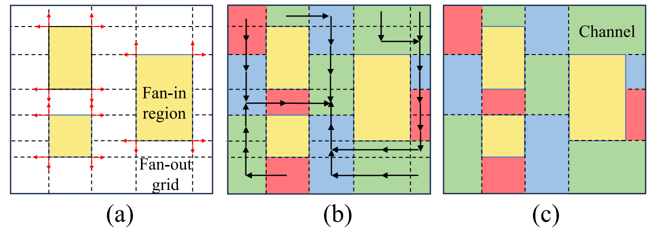

Fig. 6. Illustration of channel planning. (a) Construction of a Hanan grid [28], (b) MST generated on (a), and (c) channel planning result of (a).

III. Proposed Method

In this section, we present a hierarchical partitioning-based interchip RDL routing method for FOWLP technology. Fig. 4 shows the overall flow of the proposed method which consists of the following three major stages: 1) High-level planning; 2) hierarchical partitioning; and 3) detailed routing. The technical details of these stages are discussed below.

A. High-Level Planning

In the first stage, we present an efficient high-level planning flow for the global routing of RDLs. Our goal is to maximize the utilization of routing resources and minimize the number of conflicts caused by wire intersections.

1) Fan-in Region Identification: Since fan-in regions in an FOWLP package are usually much more congested than the fan-out regions, they need to be identified separately so that routing resources can be fully utilized during RDL routing. To this end, top vias connected to I/O pads are inserted into the top RDL to generate the chip-to-package connections as shown in Fig. 5(a), where a chip with a set of I/O pads is connected to the package using nine vias. Then, the fan-in region of a chip can be identified as the rectilinear bounding box of the top vias connected to it, and the remaining region in the layout is considered the fan-out region. For example, Fig. 5(b) shows the fan-in region of the chip described in Fig. 5(a), which is the area covered by the rectilinear bounding box of top vias t_1^v–t_9^v. Note that some top vias, e.g., t_2^v, t_4^v, and t_6^v in Fig. 5(b), may be located inside the fan-in region due to the existence of nonperipheral I/O pads. To deal with their routing design, wire segments with the shortest distances between these top vias and the boundaries of fan-in region are constructed as illustrated in Fig. 5(b), where a_{t_2}, a_{t_4}, and a_{t_6} are referred to as the access points between fan-in and fan-out regions. Note that since I/O pads are predetermined based on their real positions on the chips and any two chips cannot overlap in package, a feasible fan-in region can always be identified even if two chips are positioned very close to each other. As a consequence, an interchip net connecting two nonperipheral I/O pads can be transformed into a net connecting two access points. The computational complexity of this stage is $O(|P|)$, where $|P|$ is the number of I/O pads.

2) Channel Planning: After completing the identification of fan-in regions, a Hanan grid is constructed as the underlying graph of the global routing in the RDLs [28], [29], [30], so that resources in the given routing region can be fully utilized. Fig. 6(a) shows an example of Hanan grid construction, where the given routing region includes three fan-in regions. We project line segments from all the corners of fan-in regions at four orthogonal directions, and each line is ended when encountering a fan-in region or the boundary of the routing plane. Note that subgrids formed in the fan-out region in the Hanna grid are called *fan-out grids* as shown in Fig. 6(a).

With the constructed Hanan grid, a graph $G = (V, E)$ is then generated, where a vertex $v_i \in V$ represents a fan-out grid and an edge $e_{i,j} \in E$ if v_i and v_j are two adjacent fan-out grids in the Hanan grid. Furthermore, to avoid tiny grids that are difficult to use during routing, we employ an extended dynamic programming algorithm based on the method in [31] to merge fragmented grids in the Hanan grid. In [31], a minimum spanning tree (MST) on G is first generated, where the weight of an edge $e_{i,j} \in E$, denoted by $w_{e_{i,j}}$, is computed as

$$w_{e_{i,j}} = \Gamma\left(v_{i,j}\right) \tag{1}$$

where $v_{i,j}$ is the grid formed after merging v_i and v_j and $\Gamma(\cdot)$ is used to compute the aspect ratio of a grid. In particular, the aspect ratio of a grid $v_{i,j}$ with width w and height h can be computed as

$$\Gamma\left(v_{i,j}\right) = \max\{h/w, w/h\}. \tag{2}$$

With the generated MST, adjacent fan-out grids in the Hanan grid are then merged from leaf nodes to the root using a dynamic programming method [31]. Note that since the value of $w_{e_{i,j}}$ in (1) is mainly determined by the merged grid $v_{i,j}$ rather than grids v_i and v_j themselves, some tiny grids may still be remained in the merged Hanan grid especially when the problem scale is relatively large. To overcome this bottleneck, grids with a large aspect ratio need to be merged with a higher priority. Accordingly, we redefine the weight of an edge $e_{i,j} \in E$ as follow:

$$w'_{e_{i,j}} = \frac{\Gamma\left(v_{i,j}\right)^{\alpha}}{\Gamma(v_i) \cdot \Gamma\left(v_j\right)} \tag{3}$$

where α is a user-defined parameter. Fig. 6(b) shows the MST constructed on the Hanan in Fig. 6(a), where fan-out grids with the same color are merged sequentially to generate the channel planning result as shown in Fig. 6(c). Note that both a merged grid and a fan-in region in the final channel planning result are referred to as a *channel*.

3) Pathfinding With Dynamic Routing Graph: With the generated channel planning result, we further implement a DRG-based pathfinding method to generate optimized global routing solutions [32]. Specifically, DRG is a routing graph that can be updated dynamically in the routing plane once a

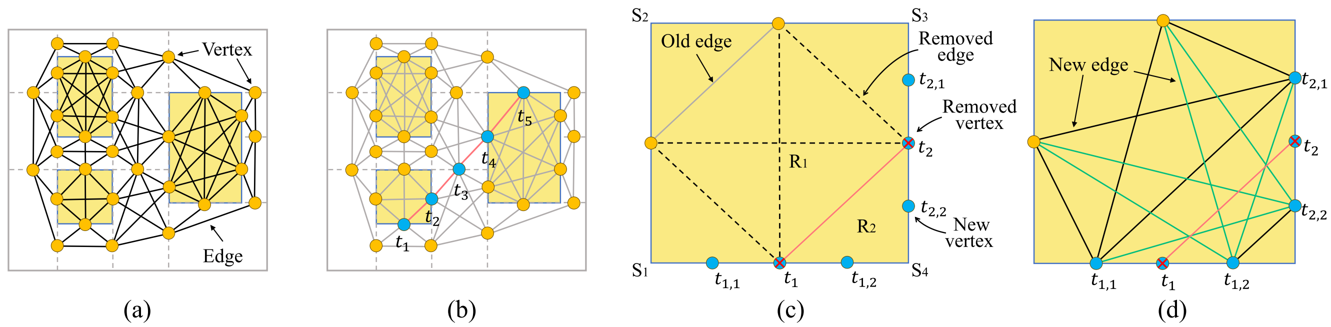

Fig. 7. Illustration of pathfinding with a DRG. (a) Initial DRG constructed based on the channel planning result described in Fig. 6(c), (b) valid routing path $\{(t_1, t_2), (t_2, t_3), (t_3, t_4), (t_4, t_5)\}$ generated on (a), (c) removal of edges/vertices and generation of new vertices regarding edge (t_1, t_2) in (b), and (d) newly added edges for the case described in (c).

Algorithm 1: Pathfinding With DRG

Input: The Initial DRG $G_0(V_0, E_0)$ and a set N of interchip nets.
Output: A global routing solution $\mathcal{S}_g$.

Initialize the weight of each edge $e(i,j) \in E_0$ according to (4);
for *each net* $n_i \in N$ **do**
  Routing path $P_i(V_i^{occ}, E_i^{occ})$=*A*_Path_Search*$(G_i, n_i)$;
  Initialize V_i^{new} and E_i^{new} to $\emptyset$;
  for *each edge* $e(j,k) \in E_i^{occ}$ **do**
    Compute newly generated vertices and add them to V_i^{new};
    for *each vertex* $v_t \in V_i^{new}$ **do**
      Compute newly generated edges and add them to E_i^{new};
      Compute the weight of each newly added edge in E_i^{new} according to (5);
    end for
  end for
  Perform $V_{i+1} = V_i - V_i^{occ} + V_i^{new}$ according to (6);
  Perform $E_{i+1} = E_i - E_i^{occ} + E_i^{new}$ according to (7);
  $G_{i+1} = (V_{i+1}, E_{i+1})$; //Graph updating
  Add P_i to $\mathcal{S}_g$;
end for

---

valid routing path is found. In particular, a valid routing path is a set of segments between channel boundaries such that the two ends of an interchip net are connected in the 2-D channel plane.

The initial DRG, denoted by $G_0(V_0, E_0)$, can be constructed based on the previously generated channel planning result, where V_0 includes all the midpoints of shared boundaries between adjacent channels. For each pair of vertices v_i and v_j in V_0, if the edge formed by v_i and v_j, denoted by $e(i,j)$, does not pass through any channel boundary, $e(i,j)$ will be added to E_0. Fig. 7(a) shows the initial DRG constructed based on the channel planning result described in Fig. 6(c). Then, the weight of an edge $e(i,j) \in E_0$ can be computed as

$$w_{e(i,j)} = Edis\big(v_i, v_j\big) \tag{4}$$

where $Edis(v_i, v_j)$ is the Euclidean distance between vertices v_i and v_j. With the constructed DRG, A* algorithm is then adopted to find a valid routing path with minimized cost for each interchip net. Note that since vias will be placed inside channels, a minimum average spacing ϵ between boundary vertices needs to be satisfied during pathfinding so that intersections between vias can be avoided, where $\epsilon = (w_e + w_v)/2 + \delta$. Fig. 7(b) shows a valid routing path generated on the DRG described in Fig. 7(a), which consists of four edges connecting the two ends t_1 and t_5 in the DRG.

On the other hand, once a valid routing path is found, DRG will be updated as mentioned before. We denote the ith $(i \geq 0)$ routing path that has been found as $P_i(V_i^{\text{occ}}, E_i^{\text{occ}})$, where V_i^{occ} and E_i^{occ} are the set of vertices and edges occupied by the path, respectively. For example, the vertices and edges occupied by the routing path shown in Fig. 7(b) are $\{t_1, t_2, t_3, t_4, t_5\}$ and $\{(t_1, t_2), (t_2, t_3), (t_3, t_4), (t_4, t_5)\}$. For each edge $e(j,k) \in E_i^{\text{occ}}$, the channel that covers $e(j,k)$ is split into two subregions. Then, four new vertices located at the midpoints of channel boundaries in each subregion can be generated. We take the edge (t_1, t_2) shown in Fig. 7(b) as an example. It can be seen from Fig. 7(c) that the channel covering edge (t_1, t_2) is split into two subregions R_1 and R_2. In R_1, new vertices $t_{1,1}$ and $t_{2,1}$ are generated, which are the midpoints between t_1 and S_1 and between t_2 and S_3, respectively. Similarly, new vertices $t_{1,2}$ and $t_{2,2}$ can be generated in the region R_2. Accordingly, we denote the set of newly generated vertices for all the edges in E_i^{occ} as $V_i^{\text{new}} = \{v_t | 0 \leq t < 2 * |V_i^{\text{occ}}|\}$.

Besides new vertices, new edges also need to be generated when a DRG is updated. Specifically, for each edge $e(j,k) \in E_i^{\text{occ}}$, after new vertices are generated, the two ends v_j and v_k as well as the edges connecting with them are first removed from DRG. Let us revisit the example shown in Fig. 7(c). After generating the four new vertices, it can be seen that vertices t_1 and t_2 and edges represented in dotted lines are removed, respectively. Then, as shown in Fig. 7(d), new edges can be added to the DRG based on the same rule for generating G_0. Similarly, we denote the set of newly added edges in respect to all the newly generated vertices as $E_i^{\text{new}} = \{e(s,t) | v_s \in V_i^{\text{new}}$ or $v_t \in V_i^{\text{new}}\}$. In particular, for an edge $e(s,t) \in E_i^{\text{new}}$, its weight can be computed as

$$w_{e(s,t)} = Edis(v_s, v_t) + z \cdot (\beta + \gamma \cdot \ln(|N|)) \tag{5}$$

where β and γ are user-defined parameters used for evaluating the influence of intersections between routing paths, $|N|$ is the number of given interchip nets, and parameter z is a 0-1 variable representing whether edge $e(s,t)$ is intersected with the original edge inside the same channel on the routing

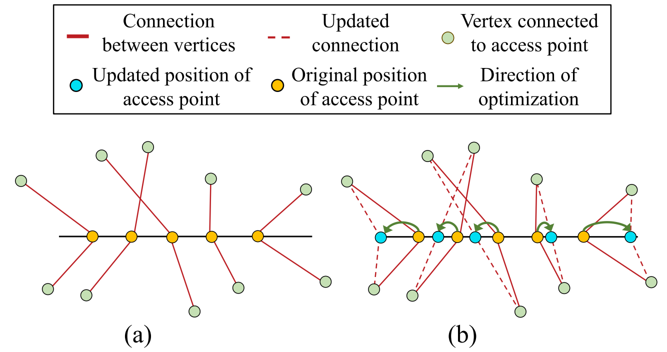

Fig. 8. Illustration of access point optimization. (a) Initial placement of a set of access points on the channel boundary and (b) optimized placement of access points in (a).

path. For example, the value of z should be set to 1 for the newly added edges in green color in Fig. 7(d) since they are intersected with the original edge (t_1, t_2). In contrast, z is set to 0 for the remaining edges in black color since they are not intersected with (t_1, t_2).

With the newly generated vertices and edges for routing path P_i, the updating of DRG, in other words the generation of the $(i+1)$th DRG, denoted by $G_{i+1}(V_{i+1}, E_{i+1})$, can be realized by updating vertices and edges in the following way:

$$V_{i+1} = V_i - V_i^{\text{occ}} + V_i^{\text{new}} \tag{6}$$

$$E_{i+1} = E_i - E_i^{\text{occ}} + E_i^{\text{new}}. \tag{7}$$

Algorithm 1 shows the pseudocode of the proposed pathfinding method with a DRG.

4) Access Point Optimization and Via Planning: With the previously generated channel planning result as well as routing paths of interchip nets, the concept of access points can now be extended to the intersection points between channel boundaries and routing paths. On the other hand, although efficient global routing solutions are already generated using the above discussed techniques, there is still another critical factor that affects the final routing quality, i.e., the positions of access points.

Since an unreasonable placement of access points can lead to unnecessary wire detours, a function monotonicity analysis-based access point optimization method is proposed to further reduce the total wirelength in this section. Before discussing the details of the proposed method, we first present a formal description of access point optimization problem: Given a set $A = \{a_i | 1 \le i \le |A|\}$ of access points located on a channel boundary, adjusting their locations such that the total Euclidean distance of their connections with other points is minimized while maintaining their topological order unchanged.

Fig. 8(a) shows a channel boundary with a set of evenly distributed access points, where each access point is connected with another two points, denoted by p_i and q_i, on the routing plane. Without loss of generality, we assume that the two ends of the channel boundary are located at $(0, 0)$ and $(L_b, 0)$, respectively. Accordingly, our objective is to minimize the total length of edges connected to access points

$$\sum_{i=0}^{|A|-1} \left(\sqrt{(x_i^a - x_i^p)^2 + (y_i^p)^2} + \sqrt{(x_i^a - x_i^q)^2 + (y_i^q)^2} \right) \tag{8}$$

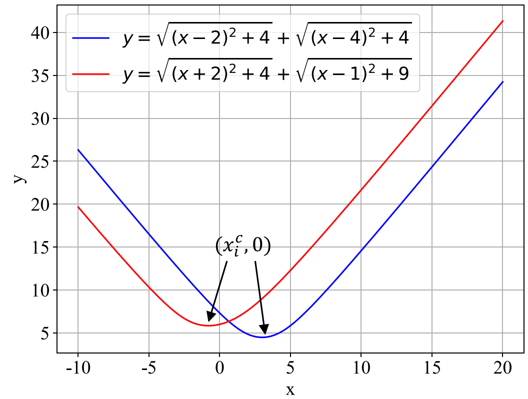

Fig. 9. Two curves of (11), where values of $(x_i^p, y_i^p, x_i^q, y_i^q)$ are set to (2, 2, 4, −2) and (−2, 2, 1, −3), respectively.

where x_i^a is the x-coordinate of access point $a_i \in A$ and (x_i^p, y_i^p) and (x_i^q, y_i^q) are the coordinates of p_i and q_i, respectively. In order to maintain the topological order of access points and satisfy the minimum spacing ϵ, the valid interval of x_i^a, denoted by $[x_i^l, x_i^r]$, can be computed as

$$x_i^l = \begin{cases} 0, & i = 0 \\ x_{i-1}^a + \epsilon, & \text{otherwise} \end{cases} \tag{9}$$

$$x_i^r = \begin{cases} L_b, & i = |A| - 1 \\ x_{i+1}^a - \epsilon, & \text{otherwise.} \end{cases} \tag{10}$$

Then, for any access point $a_i \in A$, the effects of a_i's connections on (8) can be formulated as

$$f(x_i^a) = \sqrt{\left(x_i^a - x_i^p\right)^2 + \left(y_i^p\right)^2} + \sqrt{\left(x_i^a - x_i^q\right)^2 + \left(y_i^q\right)^2}. \tag{11}$$

Furthermore, the derivative of the above function can be computed as

$$f'(x_i^a) = \frac{x_i^a - x_i^p}{\sqrt{\left(x_i^a - x_i^p\right)^2 + \left(y_i^p\right)^2}} + \frac{x_i^a - x_i^q}{\sqrt{\left(x_i^a - x_i^q\right)^2 + \left(y_i^q\right)^2}}. \tag{12}$$

Without loss of generality, we assume that x_i^p is less than x_i^q, $f'(x_i^a)$ is then always negative in $(-\infty, x_i^p)$ and positive in $(x_i^q, +\infty)$. In the interval (x_i^p, x_i^q), $f'(x_i^a)$ is the sum of two functions with opposite monotonicity, leading to a unique zero point. For example, Fig. 9 shows two concrete examples in respect to (11), where blue and red curves represent the case when $(x_i^p, y_i^p, x_i^q, y_i^q) = (2, 2, 4, -2)$ and $(x_i^p, y_i^p, x_i^q, y_i^q) = (-2, 2, 1, -3)$, respectively. It can be seen that there is a unique minimum value for both curves. As a consequence, the optimum value of x_i^a includes the following three candidates, i.e., x_i^l, x_i^c, and x_i^r, where x_i^c is the x-coordinate of the intersection between line segment (p_i, q_i) and the channel boundary. Accordingly, we choose the one that minimizes function f as the final value of x_i^a, thereby determining the optimum position of access point a_i.

Fig. 10 further shows the three types of interconnection diagrams when the optimum position of access point is located at $(x_i^l, 0)$, $(x_i^c, 0)$, and $(x_i^r, 0)$, respectively. Moreover, Fig. 8(b)

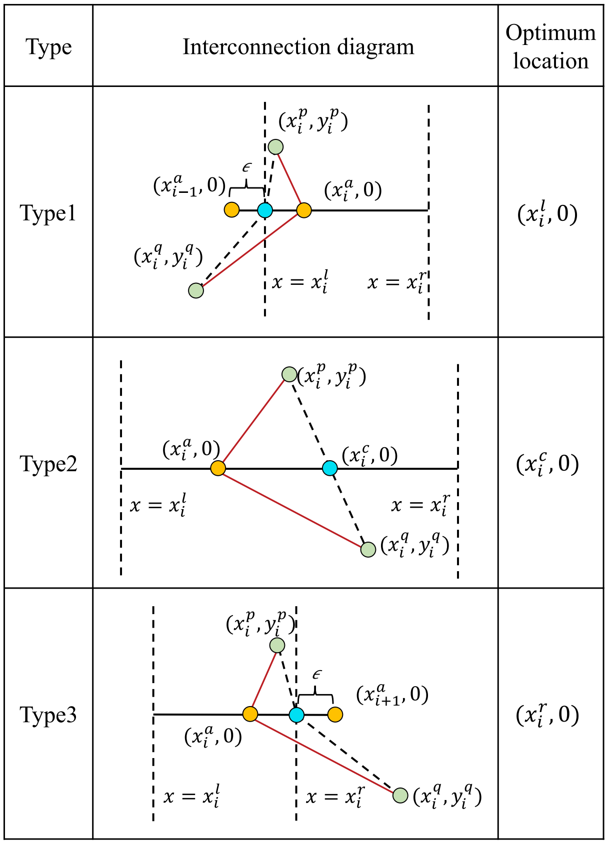

Fig. 10. Three types of interconnection diagrams when the optimum position of access point is located at $(x_i^l, 0)$, $(x_i^c, 0)$, and $(x_i^r, 0)$, respectively.

shows the optimized placement result after adjusting positions of all the access points in Fig. 8(a) using the proposed method. It can be seen that all the optimum positions in Fig. 8(b) belong to one of the three types shown in Fig. 10. Finally, Algorithm 2 shows the pseudocode of the proposed access point optimization method.

With the optimized placement result of access points, via planning can then be implemented by leveraging the flexibility of irregular vias. In the proposed method, we insert vias to the channel boundaries instead of the entire routing region, so that extra wires introduced due to via-avoidance during detailed routing can be eliminated.

In the proposed method, vias are arranged staggeredly along the access points on a channel boundary as shown in Fig. 11(a). Note that in order to avoid misconnections between irregular vias and top vias, the vertical distance between irregular vias is set to $2 * (\delta + w_v)$. Then, there is a threshold zone occupied by vias inside each channel, which cannot be used in the subsequent layer-assignment and detailed routing stages. Fig. 11(b) shows an example of wire connections inside a channel. It can be seen that layer assignment of the channel can be performed independently since there is always sufficient space in the threshold zone for vias insert.

B. Hierarchical Partitioning

With the proposed high-level planning flow, in this stage, the routing nets in each channel are rearranged to eliminate potential intersections. Moreover, layer assignment is performed using an updated DRG and an effective chord model [33].

Algorithm 2: Access Point Optimization

Input: A channel boundary B with length L_b, a set A of access points, and points p_i and q_i connected to access point $a_i \in A$.
Output: Optimized placement result of A.

```
Initialize channel boundary B by placing it horizontally
  at (0, 0) and (L_b, 0);
Initialize the access points in A by evenly distributing
  them on B;
for each a_i ∈ A do
    Computing [x_i^l, x_i^r] according to (9) and (10);
    Computing (x_i^c, 0) = Intersection_Point(B, (p_i, q_i));
    if x_i^c ∈ [x_i^l, x_i^r] then
        x_i^a ← x_i^c; //update the optimum location of a_i
    else
        d_l = Distance_Calculation(x_i^l) according to (11);
        d_r = Distance_Calculation(x_i^r) according to (11);
        if d_l < d_r then
            x_i^a ← x_i^l;
        else
            x_i^a ← x_i^r;
        end if
    end if
end for
return The placement result of A
```

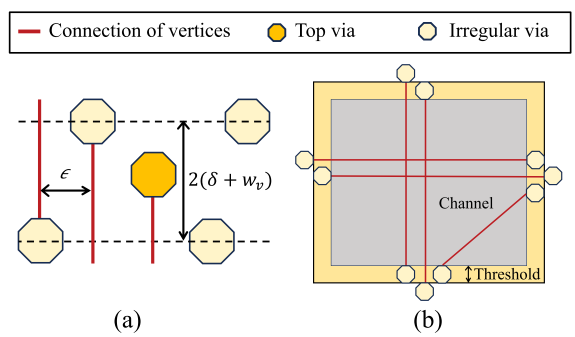

Fig. 11. Illustration of via planning. (a) Staggered via array on the channel boundary and (b) channel with nets and inserted vias.

1) Binary-Search-Based Channel Partitioning: In this step, we further partition each channel into equivalent rectangular tiles, so that conflicts between routing nets can be eliminated within tiles. For each rectangular channel, we partition it into a set of tiles with r_t rows and c_t columns. For example, the channel shown in Fig. 12(a) is divided into 12 equivalent tiles with 3 rows and 4 columns, i.e., $r_t = 3$ and $c_t = 4$. In the proposed method, values of r_t and c_t for a given channel are computed as

$$(r_t,\ c_t) = (\lfloor h_c/\lambda \rfloor,\ \lfloor w_c/\lambda \rfloor) \tag{13}$$

where w_c and h_c are the width and height of the channel, respectively. Parameter λ is used to determine the size of tiles. Note that the value of λ has a significant influence on the channel partitioning, in other words the tile segmentation result. A small value of λ can lead to larger solution space since more tile boundaries can be used to insert vias. In

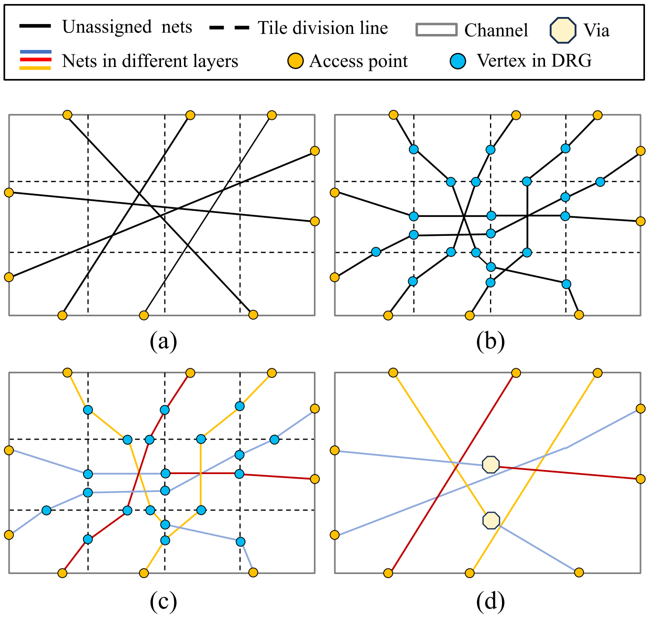

Fig. 12. Illustration of hierarchical partitioning. (a) Channel partitioning result with 12 tiles, (b) generated routing paths of nets on (a), (c) layer assignment result of (b), and (d) design solution of (c) after access point optimization and via insertion.

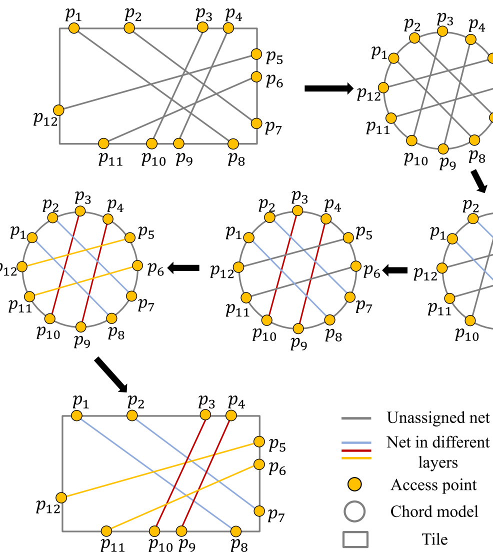

Fig. 13. Procedure of the proposed chord model-based layer assignment.

contrast, a large value of λ can reduce the total wirelength between tiles.

To realize the tradeoff between wirelength and via insert, we present a binary-search-based method to find the optimum value of λ. During the search process, for a given value of λ, we check whether tuple (r_t, c_t) can lead to a valid layer assignment result. If it fails, the upper boundary of the binary search will be shrunk. Otherwise, a valid value of λ is found, and the lower boundary of the binary search will be shrunk. This procedure is performed iteratively until the lower boundary is equal to the upper boundary, thus finding the optimum value of λ.

2) Layer Assignment Using Chord Model: Layer assignment is performed once a channel is partitioned into a set of tiles. We first construct a DRG based on the generated tiles and a feasible path connecting the two ends of each net can be generated as shown in Fig. 12(b). Then, routing nets inside each tile are assigned to appropriate layers. We map these nets to a chord model so that the maximum planar disjoint subsets (MPDSs) can be found layer by layer [33]. Once all the nets are assigned without introducing any intersection, the tile is considered solvable. Fig. 12(c) shows the layer assignment result of Fig. 12(b), where nets assigned to different layers are labeled with different colors. With the generated layer assignment result, the access point optimization and via planning methods proposed in Section III-A4) are reperformed to further reduce the total wirelength. In particular, if a net passing through adjacent tiles belongs to different layers, a via will be inserted at the crossing position between the shared tile boundary and routing path of the net as illustrated in Fig. 12(d).

Moreover, Fig. 13 shows a detailed procedure of the proposed chord model-based layer assignment. In this example, six routing nets inside a tile, i.e., $\{(p_1, p_8), (p_2, p_7), (p_3, p_{10}), (p_4, p_9), (p_5, p_{12}), (p_6, p_{11})\}$, need to be assigned to three layers. To this end, they are mapped into a circle with six chords according to their topological order. Then, we employ the method proposed in [33] to sequentially find MPDSs, where chords in each subset can be assigned to the same RDL.

C. Detailed Routing

After completing the channel planning and layer assignment, detailed routing is performed layer by layer inside each tile to generate the final RDL routing solution. Since vias are inserted on the tile boundaries, there is no conflict between wires and vias during detailed routing. We categorize the routing nets inside a tile into two types: 1) Adjacent edge-to-edge net (AEN) and 2) opposite edge-to-edge net (OEN). For an AEN, its two ends are located at the adjacent boundaries of a tile, such as the net formed by p_2 and p_7 in Fig. 13. In contrast, the two ends of an OEN, e.g., p_1 and p_8 in Fig. 13, are located at the opposite boundaries of a tile.

1) Routing of AENs: In the proposed detailed routing algorithm, AENs are first routed using a pattern routing method. Specifically, for an AEN, e.g., (p_1, p_2) shown in Fig. 14(a), a parallelogram with 45° internal angle is constructed based on the two ends of the net. Note that the feasible region that can be used for the routing of (p_1, p_2) contains the entire tile since there is only one AEN in this example. Afterward, the two edges of the parallelogram close to the center of the tile, i.e., solid lines in blue color in Fig. 14(a), form the routing path of AEN (p_1, p_2). Note that dashed lines in blue color form an invalid routing path since they violate the design rules presented in Section II-A.

To improve the routability of detailed routing, we adopt a routing strategy that deals with nets from outside to inside according to their relative positions in the tile. In other words, a higher priority will be given to the AENs close to the corners of the tile. For example, Fig. 14(b) shows the detailed routing result of a tile with six AENs, where nets (p_1, p_2), (p_3, p_4), (p_5, p_6), and (p_7, p_8) are first routed since they are close to the corners of the tile. After generating the routing paths of all

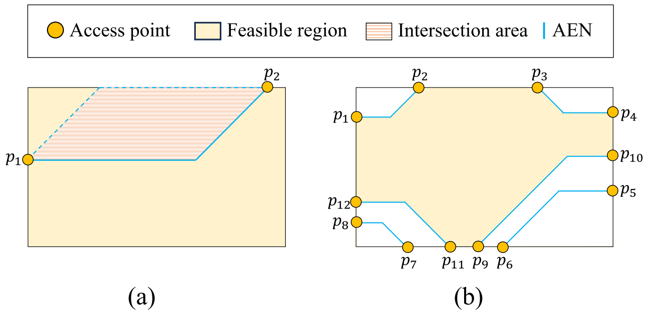

Fig. 14. (a) Illustration of pattern routing for an AEN and (b) routing result of a tile with six AENs, where the region in orange color can be further used for the routing of OENs.

the AENs, the feasible region that can be used for the routing of OENs in the tile will be updated as illustrated in Fig. 14(b).

2) Routing of OENs: After completing the routing of AENs, OENs within each tile are routed using a corner routing method. First, all the OENs in the horizontal/vertical directions are sorted in ascending order according to their positions in y/x axis. Note that because any two OENs do not intersect, there is always a certain order for OENs in either horizontal or vertical direction. Then, these OENs are dealt with sequentially by assigning a corner of the tile (also referred to as reference corner) to each of them. For an OEN, its reference corner is determined by the relative positions of its two ends and can be described as follows.

1) Left-bottom corner of the tile is assigned if the OEN is in the vertical direction and the bottom endpoint has a larger value of x-coordinate.
2) Left-top corner of the tile is assigned if the OEN is in the vertical direction and the upper endpoint has a larger value of x-coordinate.
3) Right-top corner of the tile is assigned if the OEN is in the horizontal direction and the left endpoint has a larger value of y-coordinate.
4) Right-bottom corner of the tile is assigned if the OEN is in the horizontal direction and the right endpoint has a larger value of y-coordinate.

Let us take the net (p_1, p_2) shown in Fig. 15(a) as an example, which is an OEN in the horizontal direction. Moreover, the right-bottom corner of the tile is assigned to this OEN as its reference corner. The complete procedure of corner routing can then be summarized as follows.

1) Path search is conducted along the horizontal direction from p_1 until it encounters the 45° boundary of the blockage in the reference corner direction, forming an intersection t_1 as illustrated in Fig. 15(a).
2) Path search along the 45° boundary of the blockage until it reaches the position which has the same y-coordinate with p_2, forming an intermediate point t_2 as shown in Fig. 15(b).
3) Construct path $p_1 \rightarrow t_1 \rightarrow t_2 \rightarrow p_2$ which is a valid routing path of OEN (p_1, p_2) as illustrated in Fig. 15(c).

In particular, the proposed corner routing method may lead to an invalid solution for an OEN as shown in Fig. 15(d), where the intersection is formed outside the feasible region.

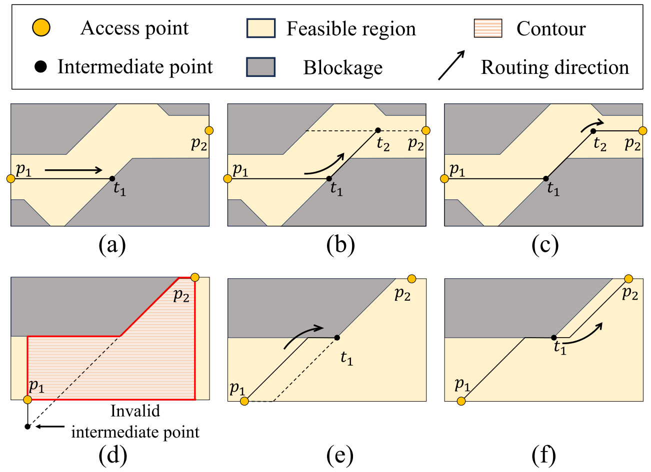

Fig. 15. Routing procedure of OENs. (a) Path search from p_1 along the horizontal direction, (b) path search along the 45° boundary of the blockage, (c) valid routing path of OEN (p_1, p_2), (d) example of an invalid intersection formed by the corner routing method, (e) routing path from p_1 to the intermediate point t_1, and (f) routing path from the intermediate point t_1 to p_2.

In this case, a contour routing method is further implemented to generate a valid routing path for the net. As shown in Fig. 15(d) and (e), the proposed contour routing includes the following four steps.

1) The contour of the OEN is first generated as illustrated in Fig. 15(d), which is the intersection area between the rectilinear bounding box of p_1 and p_2 and the original feasible region;
2) An intermediate point, e.g., t_1 in Fig. 15(d), is generated by traversing the boundary of contour clockwise;
3) As shown in Fig. 15(e) and (f), construct routing paths between endpoints of the net and the generated intermediate points using the proposed pattern routing method, thereby generating routing result of the OEN.

With the proposed detailed routing method, routing nets inside each tile in the RDLs can be routed efficiently, thus generating the final routing solution of the given interchip nets.

IV. Experimental Results

The proposed interchip RDL routing method was implemented in C++ and tested on a workstation with an Intel Xeon Gold 3.60-GHz CPU and 256-GB memory. Ten benchmark circuits are used to validate the effectiveness of proposed method as shown in Table I, where $RDL_{16}01$, $RDL_{16}02$, $RDL_{16}03$, and $RDL_{16}05$ are benchmarks reported in [16], $RDL_{17}01$, $RDL_{17}02$, and $RDL_{17}03$ are benchmarks reported in [17], $RDL_{co}04$ is a benchmark used in both [16] and [17], and $RDL_{ex}01$ and $RDL_{ex}02$ are two generated large circuits. These benchmarks are very dense and congested designs with predefined interchip nets and pads. They are generated based on real applications using the FLOWLP technology and thus have a similar level of complexity to real designs. Among them, the most complicated test case $RDL_{ex}02$ includes more than 12 000 flightline collisions. This implies that each interchip net would potentially have nearly 40 net shorts in

TABLE I
DETAILS OF THE BENCHMARKS USED IN THE EXPERIMENTS[3]

Benchmarks (Area, #Chips, $\|P\|$, $\|N\|$, $\|L\|$, w_v, w_e, δ)	
$RDL_{16}01$ [16] (5×5, 2, 44, 22, 2, 20, 4, 4)	$RDL_{16}02$ [16] (6×4, 5, 158, 79, 3, 20, 4, 4)
$RDL_{16}03$ [16] (10×5, 6, 222, 111, 3, 20, 4, 4)	$RDL_{16}05$ [16] (10×10, 9, 522, 261, 4, 20, 2, 2)
$RDL_{17}01$ [17] (5×5, 2, 44, 22, 4, 20, 4, 4)	$RDL_{17}02$ [17] (10×5, 6, 222, 111, 5, 20, 4, 4)
$RDL_{17}03$ [17] (10×10, 9, 522, 261, 5, 20, 2, 2)	$RDL_{co}04$ [16] [17] (7.5×7.5, 3, 92, 46, 3, 20, 4, 4)
$RDL_{ex}01$ (16×12, 20, 458, 229, 2, 20, 2, 4)	$RDL_{ex}02$ (16×12, 20, 618, 309, 4, 20, 2, 2)

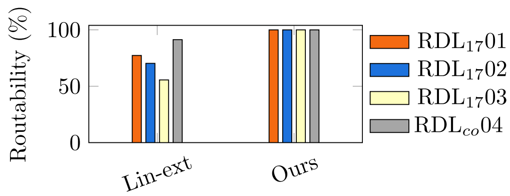

Fig. 16. Comparison with Lin-ext [34] regarding the percentage of routability.

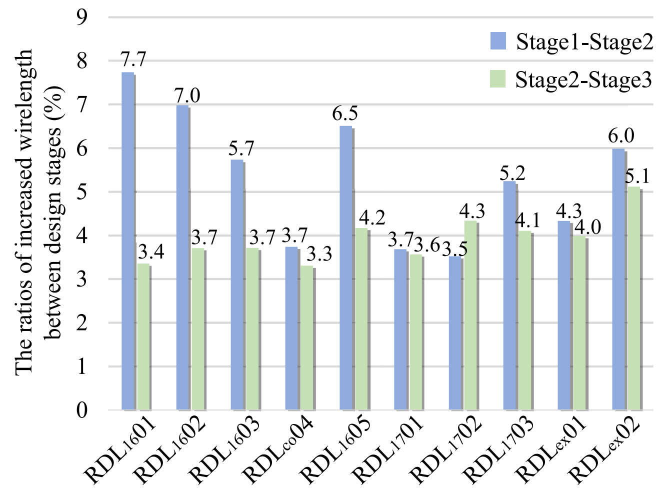

Fig. 17. Increased wirelength between design stages.

the same layer if designed improperly. Even if the nets are uniformly assigned to different RDLs, there are still more than 2 400 flightline collisions. The details of these benchmarks are reported in Table I, where Area, #Chips, $|P|$, $|N|$, $|L|$, w_v, w_e, and δ represent the area of the routing region in each layer, the number of chips, the number of I/O pads, the number of interchip nets, the number of routing layers, the width of vias, the width of wires, and the minimum spacing between wires/vias, respectively. In addition, parameter α used for computing the weight of an edge in Formula (3) is set to 1.5. Parameters β and γ used for computing the weights of DRG edges in Formula (5) are set to -1300 amd 400, respectively.

First, to evaluate the overall performance of the proposed method, we implemented a channel-guided RDL router called CRouter based on the PCB planning method proposed in [32]. CRouter first generates channel divisions using a Hanan grid-based approach and insert vias to the RDLs before global routing. Then, both layer assignment and detailed routing are performed by formulating them into an integer LP model to generate the final routing solution. Table II shows the comparison results between the proposed method and CRouter regarding the routability, the total wirelength, the maximum wirelength, and the CPU time, where *Imp* provides the relative improvement of our method over CRouter. It can be seen that CRrouter fails to complete the routing of the given nets on three benchmarks, including $RDL_{16}05$, $RDL_{ex}01$, and $RDL_{ex}02$. In contrast, the proposed method achieves 100% routability across all the benchmarks with an average reduction rate of 8.40% in terms of the total wirelength. This is mainly because our method adopts a hierarchical partitioning-based routing strategy and inserts vias after pathfinding, and thus can deal with circuits with a very high density. Furthermore, compared with CRouter, the proposed method achieves a 4.68%-23.39% reduction on the maximum wirelength with an average reduction rate of 11.86%, thus demonstrating its performance in respect to the worst-case signal integrity. We provide the runtime of the two methods in the last two columns of Table II. It can be seen that the proposed method is 2.19 times faster than CRouter.

Also, Lin-ext [34] is a concurrent RDL routing method using fixed vias and regular pads. Particularly, it employs an A*-search-based sequential routing strategy to enhance routability. To verify the advantages of flexible vias/irregular pads used in the proposed method, we further make comparison between the two methods based on the results reported in [17]. It can be seen from Fig. 16 that Lin-ext cannot realize 100% routability in the benchmarks due to the adoption of fixed vias. In contrast, all the given interchip nets can be routed successfully by the proposed method, demonstrating the vast flexibility of the proposed method.

To further validate the effectiveness of the proposed method, we compare it with the algorithms proposed in [16] and [17], which are the most recent techniques that deal with the interchip RDL routing considering both flexible vias and irregular pads in FOWLP technology. Reference [16] employs a Voronoi diagram-based method and a chord-based tile model to handle the placement of irregular vias and the routing of interchip nets, respectively, so that RDL routing solutions with better routability and shorter wirelength can be generated efficiently. Table III shows the comparison results between our method and [16]. It can be seen that both methods achieve 100% routability in the benchmarks, demonstrating their effectiveness in dealing with circuits with varied scales and complexities. Moreover, the proposed method achieves a 4.90%–13.21% wirelength reduction across the benchmarks, with an average reduction rate of 8.68%. This is mainly because irregular vias in [16] are placed within tiles before detailed routing. To avoid intersections between wires and vias, a large number of detours have to be introduced in the routing stage, thus significantly increasing the total wirelength.

[3]The unit of area is $10^3\mu$m × $10^3\mu$m; The unit of $w_v/w_e/\delta$ is μm.

TABLE II
COMPARISON WITH CROUTER REGARDING THE PERCENTAGE OF ROUTABILITY, THE TOTAL WIRELENGTH, THE MAXIMUM WIRELENGTH, AND THE CPU TIME

Benchmark	Routability (%)		Total wirelength (μm)			Maximum wirelength (μm)			CPU time (sec.)	
	CRouter	Ours	CRouter	Ours	Imp (%)	CRouter	Ours	Imp (%)	CRouter	Ours
RDL$_{16}$01	100	100	87303	82445	5.89	8843	6853	22.50	5.93	0.93
RDL$_{16}$02	100	100	252223	242358	4.07	10055	9446	6.06	30.24	5.40
RDL$_{16}$03	100	100	491284	475437	3.33	13211	12135	8.14	53.88	34.14
RDL$_{co}$04	100	100	228750	205061	11.56	7345	7001	4.68	9.19	2.86
RDL$_{16}$05	97.70	100	>1733790	1616681	-	-	14425	-	661.28	341.04
RDL$_{17}$01	100	100	91977	79147	16.21	8660	6634	23.39	3.51	0.43
RDL$_{17}$02	100	100	505238	467577	8.05	12897	11375	11.80	165.33	41.35
RDL$_{17}$03	100	100	1787641	1593894	12.16	14773	13822	6.44	773.44	402.24
RDL$_{ex}$01	97.82	100	>553766	535421	-	-	9247	-	492.17	174.33
RDL$_{ex}$02	90.61	100	>1120355	1134318	-	-	15311	-	833.15	381.19
Average	98.61	100	-	-	8.40	-	-	11.86	2.19	1.00

TABLE III
COMPARISON WITH [16] REGARDING THE PERCENTAGE OF ROUTABILITY, THE TOTAL WIRELENGTH, AND THE CPU TIME

Benchmark	Routability (%)		Total wirelength (μm)			CPU time (sec.)	
	Cai [16]	Ours	Cai [16]	Ours	Imp (%)	Cai [16]	Ours
RDL$_{16}$01	100	100	86695	82445	4.90	0.79	0.93
RDL$_{16}$02	100	100	258172	242358	6.12	3.92	5.40
RDL$_{16}$03	100	100	512619	475437	7.25	9.13	34.14
RDL$_{co}$04	100	100	236269	205061	13.21	0.60	2.86
RDL$_{16}$05	100	100	1835410	1616681	11.92	12.5	341.04
Average	100	100	-	-	8.68	-	-

TABLE IV
COMPARISON WITH [17] REGARDING THE PERCENTAGE OF ROUTABILITY, THE TOTAL WIRELENGTH, AND THE CPU TIME

Benchmark	Routability (%)		Total wirelength (μm)			CPU time (sec.)	
	Wen [17]	Ours	Wen [17]	Ours	Imp (%)	Wen [17]	Ours
RDL$_{17}$01	100	100	81302	79147	2.65	8.60	0.43
RDL$_{17}$02	100	100	495255	467577	5.59	56.60	41.35
RDL$_{17}$03	100	100	1776860	1593894	10.30	627.10	402.24
RDL$_{co}$04	100	100	206816	205061	0.85	36.30	2.86
Average	100	100	-	-	4.85	-	-

TABLE V
RUNTIME BREAKDOWN OF EACH STAGE IN THE PROPOSED METHOD

Benchmark	High-level planning runtime (sec.)	Ratio of stage1 (%)	Hierarchical partitioning runtime (sec.)	Ratio of stage2 (%)	Detailed routing runtime (sec.)	Ratio of stage3 (%)	Total runtime (sec.)
RDL$_{16}$01	0.03	3.22	0.88	94.62	0.02	2.16	0.93
RDL$_{16}$02	0.16	2.96	5.18	95.93	0.06	1.11	5.40
RDL$_{16}$03	12.01	35.18	22.03	64.53	0.10	0.29	34.14
RDL$_{co}$04	0.09	3.15	2.73	95.46	0.04	1.39	2.86
RDL$_{16}$05	33.00	9.67	307.75	90.24	0.29	0.09	341.04
RDL$_{17}$01	0.03	6.98	0.38	88.38	0.02	4.64	0.43
RDL$_{17}$02	31.02	75.02	10.26	24.81	0.07	0.17	41.35
RDL$_{17}$03	33.00	8.20	368.98	91.73	0.26	0.07	402.24
RDL$_{ex}$01	3.93	2.25	169.58	97.28	0.82	0.47	174.33
RDL$_{ex}$02	14.75	3.87	365.19	95.8	1.25	0.33	381.19
Average	-	15.05	-	83.88	-	1.07	-

The method in [17] adopts a weighted-MPSC-based layer assignment algorithm to carry out concurrent intralayer routing. Then, an LP-based layout optimization algorithm is implemented to further realize via arrangement and wirelength reduction. The comparison results between our method and [17] are reported in Table IV. Similarly, due to the strong robustness of the two methods, they realize 100% routability across all the benchmarks. However, owing to the lack of an overall planning of the routing region, [17] generates many tiny fragment tiles during routing, leading to some redundant vias and extra wire detours in the final routing solutions. Accordingly, it can be seen from Table IV that the proposed

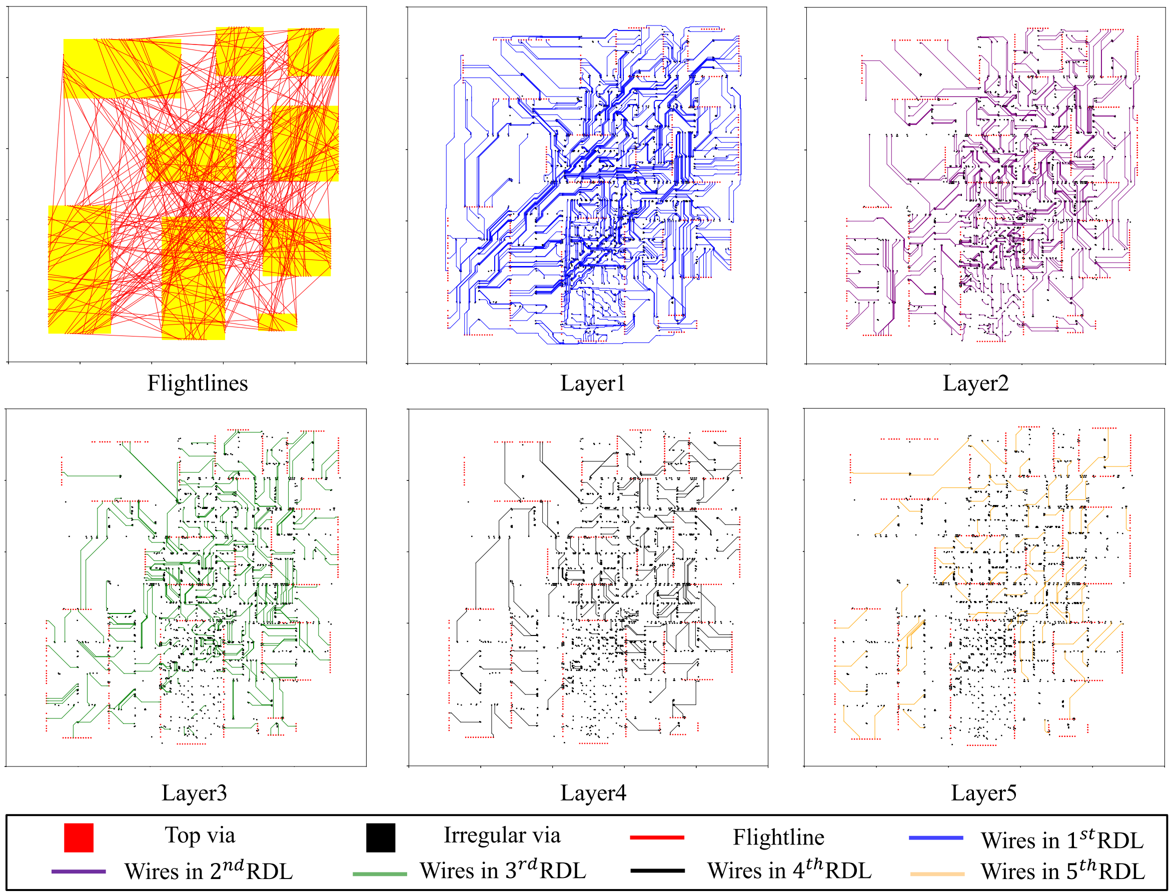

Fig. 18. Pin-connection flightlines and the resulting 5-layer RDL solution of benchmark $\mathrm{RDL}_{17}03$.

method achieves a 0.85%–10.30% wirelength reduction in the benchmarks, with an average reduction rate of 4.85%, thereby demonstrating its effectiveness in wirelength reduction. Note that the method in [17] achieves a comparable result in respect to case $\mathrm{RDL}_{co}04$ due to a strong exploration capability of the LP-based optimization algorithm. Such a result is possible when the scale of the circuit is not very large. Nonetheless, for large-scale circuits, such as $\mathrm{RDL}_{17}03$, the wirelength reduction of the proposed method reaches up to 10.30%, further demonstrating its effectiveness in dealing with large-scale design.

In addition, the last two columns of Tables III and IV show the CPU times of the three methods. It can be seen that the proposed method generates the best routing solutions on average within a short period of time. Table V further shows the runtime breakdown of each design stage in the proposed method. It can be seen that high-level planning and hierarchical partitioning account for 15.05% and 83.88% of the total runtime on average, respectively. Due to the global optimization of the first two stages, such as channel planning/partitioning, layer assignment, access point/via planning, etc., detailed routing solutions with minimized wirelength can be generated in a highly efficient manner as illustrated in columns 6 and 7 of Table V.

The reasons why the proposed method achieves better RDL routing solutions are analyzed as follows.

1) Fragment channels that can lead to wire detours are avoided as much as possible by the proposed channel planning method, thus improving the overall utilization of routing resources.
2) Since intersections between routing nets are allowed during routing region planning, extra wirelength due to intersection avoidance can be reduced.
3) The number of tiles is reduced significantly due to the adoption of a binary search-based channel partitioning method, thus significantly reducing the number of wire detours.
4) Due to the systematic optimization of access point locations, wirelengths between channels/tiles are reduced efficiently.
5) Since vias are placed at the boundary of tiles, extra wires introduced for via-avoidance in the detailed routing can be avoided.

Overall, with the aforementioned optimization techniques, design stages in the proposed method are seamlessly integrated to complete different tasks in an efficient manner, resulting in RDL solutions with 100% routability and minimized wirelength. On the other hand, although problem complexities are increased as the design stages move forward, the ratios of increased wirelength between design stages are reduced as illustrated in Fig. 17, further demonstrating the global optimization capability of the proposed method. Finally, Fig. 18 shows the pin-connection flightlines and the final routing solution of circuit $\mathrm{RDL}_{17}03$, which is a very congested test case in the benchmarks. In $\mathrm{RDL}_{17}03$, nine chips with 522 I/O pads and 261 interchip nets need to be connected using five RDLs within a $10\,000\mu\text{m} \times 10{,}000\mu\text{m}$ routing region. Besides, since the total number of flightline collisions reaches up to 10 245 in $\text{RDL}_{17}03$, it is very difficult to route if the routing resources are not fully utilized. With the proposed method, it can be seen from Fig. 18 that all the interchip nets are distributed evenly to different layers and the routing network in each layer is constructed successfully without violating any design rules.

V. Conclusion

We have investigated the interchip redistribution layer routing problem in modern FOWLP considering flexible vias and irregular pads simultaneously. A hierarchical partitioning-based algorithm is proposed to solve this problem. With the proposed method, optimized routing region partition can be generated and a DRG is constructed for efficient global routing. To improve routability in RDLs, a tile segmentation method with binary search is used to efficiently remove net conflicts. Moreover, wire detours are further reduced using a staggered via planning strategy. Experimental results on multiple benchmarks have confirmed that the proposed algorithm achieves better solutions compared with the state-of-the-art techniques. Signal integrity optimization in the FOWLP technology will be considered in our future work.

References

[1] L. Lee, L. Kang, Y. T. Kwon, T. Kim, E. Lee, and J. Lee, "FOWLP technology as wafer level system in packaging (SiP) solution," in *Proc. ICEP*, 2017, pp. 491–493.

[2] L. T. Guan, D. Ho, J. M. Chinq, and S. Bhattacharya, "FOWLP design for HBM applications," in *Proc. IEEE ECTC*, 2017, pp. 1–4.

[3] C.-T. Wang and D. Yu, "Power-performance advantages of InFO technology for advanced system integration," in *Proc. 3DIC*, 2019, pp. 1–4.

[4] A. Cardoso, M. Pires, R. Pinto, and G. Hantos, "Thermally enhanced FOWLP-development of a power-eWLB demonstrator," in *Proc. IEEE ECTC*, 2015, pp. 1682–1688.

[5] D. Yang and R. Fontaine (Nepean, ON, Canada). *The Samsung Galaxy S23 Ultra-Inside the Flagship*. Accessed: Nov. 1, 2023. [Online]. Available: https://www.techinsights.com/blog/samsung-galaxy-s20-teardown-analysis.

[6] D. McGrath, "Teardown slideshow: Inside the third-generation iPad," Accessed: Nov. 1, 2023. [Online]. Available: https://www.eetimes.com/teardown-slideshow-inside-the-third-generation-ipad.

[7] C. C. Liu et al., "High-performance integrated fan-out wafer level packaging (InFO-WLP): Technology and system integration," in *Proc. IEEE IEDM*, 2012, pp. 1–4.

[8] H. Chen, H.-C. Lin, C.-N. Peng, and M.-J. Wang, "Wafer level chip scale package copper pillar probing," in *Proc. ICT*, 2014, pp. 1–6.

[9] S.-M. Chen et al., "High performance heterogeneous integration on fan-out RDL interposer," in *Proc. Symp. VLSI Technol.*, 2019, pp. 52–53.

[10] X. Hua et al., "Development of chip-first and die-up fan-out wafer level packaging," in *Proc. ECTC*, 2017, pp. 1–6.

[11] C.-F. Tseng, C.-S. Liu, C.-H. We, and D. Yu, "InFO (wafer level integrated fan-out) technology," in *Proc. IEEE ECTC*, 2016, pp. 1–6.

[12] S.-H. You et al., "Advanced fan-out package SI/PI/thermal performance analysis of novel RDL packages," in *Proc. IEEE ECTC*, 2018, pp. 1295–1301.

[13] C.-T. Wang et al., "Signal integrity of submicron InFO heterogeneous integration for high-performance computing applications," in *Proc. IEEE ECTC*, 2019, pp. 688–694.

[14] N.-C. Chen et al., "A novel system in package with fan-out WLP for high speed SERDES application," in *Proc. IEEE ECTC*, 2016, pp. 1495–1501.

[15] T. Y. Ouyang et al., "Fan-out wafer-level packaging process integration for artificial intelligence real dies," in *Proc. IMPACT*, 2023, pp. 290–293.

[16] Y.-J. Cai, Y. Hsu, and Y.-W. Chang, "Simultaneous pre-and free-assignment routing for multiple redistribution layers with irregular vias," in *Proc. ACM/IEEE DAC*, 2021, pp. 1147–1152.

[17] H.-T. Wen, Y.-J. Cai, and Y.-W. Chang, "Via-based redistribution layer routing for InFO packages with irregular pad structures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5554–5567, Dec. 2022.

[18] J.-C. Souriau, O. Lignier, M. Charrier, and G. Poupon, "Wafer level processing of 3D system in package for RF and data application," in *Proc. IEEE ECTC*, 2005, pp. 356–361.

[19] (IEEE Electron. Packag. Soc., Piscataway, NJ, USA). *Heterogeneous Integration Roadmap*. Accessed: May 9, 2024. [Online]. Available: https://eps.ieee.org/technology/heterogeneous-integration-roadmap.html.

[20] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang, "An integer-linear-programming-based routing algorithm for flip-chip designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 1, pp. 98–110, Jan. 2009.

[21] C.-W. Lin, P.-W. Lee, Y.-W. Chang, C.-F. Shen, and W.-C. Tseng, "An efficient pre-assignment routing algorithm for flip-chip designs," *IEEE TCAD*, vol. 31, no. 6, pp. 878–889, Jun. 2012.

[22] H. Chen et al., "Reinforcement learning guided detailed routing for custom circuits," in *Proc. ISPD*, 2023, pp. 26–34.

[23] Y. Lin, T. Qu, Z. Lu, Y. Su, and Y. Wei, "Asynchronous reinforcement learning framework and knowledge transfer for net-order exploration in detailed routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 9, pp. 3132–3142, Sep. 2022.

[24] J. Chen et al., "GPCB routing: Generative pre-trained transformers based printed circuit board routing method," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 44, no. 4, pp. 1420–1433, Apr. 202.

[25] H.-P. Pu, H. Kuo, C. Liu, and C. Douglas, "A novel submicron polymer re-distribution layer technology for advanced info packaging," in *Proc. IEEE ECTC*, 2018, pp. 45–51.

[26] T.-C. Lin, C.-C. Chi, and Y.-W. Chang, "Redistribution layer routing for wafer-level integrated fan-out package-on-packages," in *Proc. IEEE ICCAD*, 2017, pp. 561–568.

[27] T.-C. Lin, D. Merrill, Y.-Y. Wu, C. Holtz, and C.-K. Cheng, "A unified printed circuit board routing algorithm with complicated constraints and differential pairs," in *Proc. ASPDAC*, 2021, pp. 170–175.

[28] M. Hanan, "On Steiner's problem with rectilinear distance," *SIAM J. Appl. Math.*, vol. 14, no. 2, pp. 255–265, 1966.

[29] W. Guo and X. Huang, "PORA: A Physarum-inspired obstacle-avoiding routing algorithm for integrated circuit design," *Appl. Math. Model.*, vol. 78, pp. 268–286, Feb. 2020.

[30] T. Huang and E. F. Young, "Obstacle-avoiding rectilinear Steiner minimum tree construction: An optimal approach," in *Proc. IEEE/ACM ICCAD*, 2010, pp. 610–613.

[31] Y.-K. Ho et al., "Obstacle-avoiding free-assignment routing for flip-chip designs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 33, no. 2, pp. 224–236, Feb. 2014.

[32] P.-C. Wu, Q. Ma, and M. D. F. Wong, "An ILP-based automatic bus planner for dense PCBs," in *Proc. ASPDAC*, 2013, pp. 181–186.

[33] K.-J. Supowit, "Finding a maximum planar subset of a set of nets in a channel," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 6, no. 1, pp. 93–94, Jan. 1987.

[34] B.-Q. Lin, T.-C. Lin, and Y.-W. Chang, "Redistribution layer routing for integrated fan-out wafer-level chip-scale packages," in *Proc. IEEE ICCAD*, 2016, pp. 1–6.

Haoyang Xu is currently pursuing the undergraduate degree with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China.

He will pursue the Ph.D. degree with the Chinese University of Hong Kong, Hong Kong. His current research interest include electronic design automation (EDA), especially EDA for 2.5-D/3-D packaging.

Mr. Xu was a recipient of 2 gold medals in the International Collegiate Programming Contests, Asia Regions. He was also a finalist of the ICPC World Finals in 2023 and 2024.

Xing Huang (Senior Member, IEEE) received the Ph.D. degree in electronic science and technology from Fuzhou University, Fuzhou, China, in 2018.

He is currently a Professor with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His current research interests include design automation for integrated circuits and microfluidic biochips.

Dr. Huang is a recipient of the MOST International Fellowship from the Ministry of Science and Technology of Taiwan Province in 2018; the TUM University Foundation Fellowship from the Technical University of Munich, Munich, Germany in 2019; and the Humboldt Research Fellowship from the Alexander von Humboldt Foundation in 2020. He has served on the Technical Program Committees of several conferences, such as DAC, ICCAD, and ASP-DAC.

Zhen Zhuang received the M.Eng. and B.Eng. degrees from Fuzhou University, Fuzhou, China, in 2021 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Chinese University of Hong Kong, Hong Kong.

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

Mr. Zhuang was a recipient of 3 ICCAD/ISPD contest awards.

Zhiwen Yu (Senior Member, IEEE) received the Ph.D. degree in computer science from Northwestern Polytechnical University (NPU), Xi'an, China, in 2005.

He is currently a Professor with the School of Computer Science, NPU. He is also a Vice President with Harbin Engineering University, Harbin, China. His research interests include mobile crowd sensing and sensing chip design.

Bing Guo (Senior Member, IEEE) received the Ph.D. degree in computer science from Keio University, Tokyo, Japan, in 2009.

He was a Postdoctoral Researcher with Institut Telecom SudParis, Paris, France, from 2009 to 2011. He is currently a Professor with the School of Computer Science, Northwestern Polytechnical University, Xi'an, China. His research interests include mobile crowd sensing and sensing chip design.

Kai-Yuan Chao received the M.S.E. and Ph.D. degrees in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA, in 1992 and 1995, respectively.

He has worked with Synopsys as the RD Group Director for Technology Pathfinding in Strategic Programs. He also advised few successful companies in AI software, EDA, micro-LED, integrated voltage regulator, and thermal solution. Since 2024, He has been with Siemens EDA. His research interests include 3DIC, intelligent mass transfer applications, cross-domain technology co-optimization, and design automation.

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

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

Dr. Yu received 11 Best Paper Awards from ICCAD 2024, 2021, and 2013, IEEE TSM 2022, DATE 2022, ASPDAC 2021 and 2012, ICTAI 2019, Integration, the VLSI Journal in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, 6 ICCAD/ISPD contest awards, IEEE CEDA Ernest S. Kuh Early Career Award in 2021, DAC Under-40 Innovator Award in 2024, and Hong Kong RGC Research Fellowship Scheme Award in 2024. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees.

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

He is a Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. His research interests include design automation and test for microfluidic biochips.

Dr. Ho was a recipient of the Best Paper Awards at the VLSI Test Symposium in 2013 and IEEE TCAD in 2015. He currently serves as Associate Editor for the IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, *ACM Transactions on Design Automation of Electronic Systems*, *ACM Journal on Emerging Technologies in Computing Systems*, *ACM Transactions on Embedded Computing Systems*, and IEEE Transactions on Very Large Scale Integration (VLSI) Systems, and the Technical Program Committees of DAC, ICCAD, DATE, ASP-DAC.

Martin D. F. Wong (Life Fellow, IEEE) received the B.Sc. degree in mathematics from the University of Toronto, Toronto, ON, Canada, the M.S. degree in mathematics and the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign (UIUC), Champaign, IL, USA.

He was Bruton Centennial Professor of Computer Science with UT-Austin, Austin, TX, USA and Edward C. Jordan Professor of ECE with UIUC. From August 2012 to December 2018, he was the Executive Associate Dean of the College of Engineering with UIUC. From January 2019 to August 2023, he was the Dean of Engineering and Choh-Ming Li Professor of Computer Science and Engineering with The Chinese University of Hong Kong, Hong Kong. Since August 2023, he joined Hong Kong Baptist University (HKBU), Hong Kong, as the Provost of HKBU and a Chair Professor of Computer Science. He has published around 500 papers and graduated over 50 Ph.D. students in EDA. His main research interest is in electronic design automation.

Dr. Wong is a Fellow ACM.