Via-based Redistribution Layer Routing for InFO Packages with Irregular Pad Structures

Hsiang-Ting Wen[1], Yu-Jie Cai[1], Yang Hsu[1], and Yao-Wen Chang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 10617, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan
hstwen@eda.ee.ntu.edu.tw; jerrytsai@eda.ee.ntu.edu.tw; yhsu@eda.ee.ntu.edu.tw; ywchang@ntu.edu.tw

***Abstract*—The integrated fan-out (InFO) wafer-level chip-scale package is introduced for modern system-in-package designs with larger I/O counts and higher interconnection density. A redistribution layer (RDL) in an InFO package is an extra metal layer for inter-chip connections. To achieve flexible and compact inter-chip connections, the RDL routing problem for InFO packages has become a crucial problem for modern electronic designs. In advanced high-density InFO packages, multiple RDLs with flexible vias are often adopted. On the other hand, to integrate chips of different technology nodes into one package, irregular pad structures need to be considered. To our best knowledge, however, there is no published work for RDL routing considering flexible vias or irregular pad structures. In this paper, we present the first work to handle the routing problem with pre-assigned pad pairs (i.e., the hardest pre-assignment routing problem) on the via-based multi-chip multi-layer InFO package with irregular pad structures. We first propose a layer assignment method based on a weighted maximum planar subset of chords algorithm to concurrently route as many inter-chip nets as possible. We then propose an octagonal tile model with a layout partitioning method to tackle increasingly popular irregular structures. Finally, we develop an efficient linear-programming-based layout optimization algorithm to find solutions with high-quality wirelength and via arrangements. Experimental results demonstrate the effectiveness and robustness of our algorithm.**

I. Introduction

As the system design complexity grows dramatically, potential solutions for heterogeneous system integration have attracted more and more attention. Among those solutions, the *system-in-package (SiP) technology* provides low costs, small form factors, and flexible integration for complicated systems such as the Apple Watch series [1]. Specifically, TSMC's *integrated fan-out (InFO) wafer-level chip-scale package (WLCSP) technology* [2] has shown its advantages over other advanced packaging technologies. In an InFO package, heterogeneous chips can be integrated, and the inter-chip connections can be completed by extra metal layers called *redistribution layers (RDLs)*.

Figure 1(a) shows a cross-sectional view of a multi-chip multi-layer InFO WLCSP. On the top of the package, there is a molding compound covering multiple chips and passive devices, forming a base for the package and also provides physical protection. Beneath the molding compound, there are RDLs composed of alternating via layers and wire layers. A via layer provides inter-layer interconnections by RDL vias, and a wire layer provides intra-layer interconnections by metal wires. As shown in Figure 1(b), the RDL structure can be divided into two parts: (1) the fan-in region, which is the shaded area of the chips or passive devices, and (2) the fan-out region, which is the area not belonging to the fan-in region. Through these RDLs, signals can be transmitted between I/O pads of different chips (an inter-chip connection) or between an I/O pad and a bump pad (a chip-to-board connection). Finally, bump pads and micro bump balls would connect to a PCB.

A. Previous Works

The development of flip-chip routing is surveyed comprehensively in [3]. In short, a flip-chip routing problem can be classified by the flip-chip structure and its pad assignment method. For flip-chip structures, there are two types: (1) *peripheral-I/O structure* where the I/O pads are placed along the flip-chip periphery, and (2) *area-I/O structure* where I/O pads are placed everywhere in the flip chip. For pad assignment, there

This work was partially supported by AnaGlobe, Maxeda, Synopsys, TSMC, MOST of Taiwan under Grant MOST 106-2221-E-002-203-MY3, MOST 107-2221-E-002-161-MY3, MOST 108-2911-I-002-544, and MOST 108-2221-E-002-097-MY3.

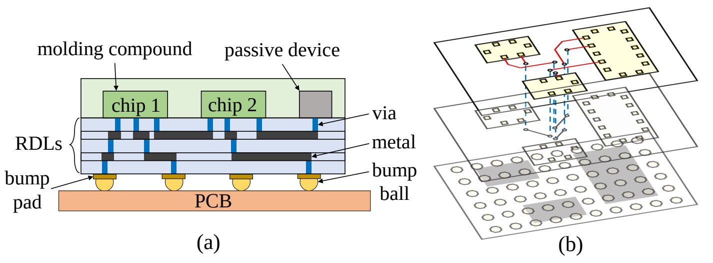

Fig. 1: (a) A cross-sectional view of a via-based multi-chip multi-layer InFO WLCSP. (b) The RDL structure of a via-based multi-chip multi-layer InFO package with three RDLs.

are three types: (1) *free-assignment (FA) routing*, (2) *pre-assignment (PA) routing*, and (3) *unified-assignment (UA) routing*.

For FA routing, the net assignments between I/O pads and bump pads are not predefined before routing, so a router has the freedom to assign each I/O pad to an arbitrary bump pad. In contrast, for PA routing, connections between I/O pads and bump pads are predefined before routing. PA routing is harder than FA one because the mapping between I/O pads and bump pads impose more constraints. For UA routing, some net assignments are predefined, and some are not. As a result, both the FA and PA routing problems need to be considered to achieve better solutions.

1) Flip-chip Routing: In general, the FA routing problem can be solved by *network-flow algorithms* [4], [5] under well-defined tile models [6], [7]. Fang *et al.* [4] first addressed the FA routing problem and applied *a minimum-cost maximum-flow (MCMF) algorithm* to generate a routing result on the peripheral I/O structure. Fang and Chang [5] developed a two-stage algorithm to handle the FA routing problem in chip-package co-design on the area-I/O structure with multiple RDLs without vias. The two-stage flow consists of an MCMF-based global routing stage and a detailed routing stage.

For the PA routing problem, various algorithms for different pad structures were proposed. Fang *et al.* [8] first proposed an integer-linear-programming (ILP)-based routing algorithm to solve the PA routing problem on the peripheral-I/O structure. They also developed an effective reduction technique to prune the ILP variables. Lin *et al.* [9] developed a more efficient approach by computing the *longest common subsequence (LCS)* and the *maximum planar subset of chords (MPSC)* to achieve significant speedups compared with [8].

For the UA routing problem, Fang *et al.* [10] considered the *area-I/O structure*. To tackle with the area-I/O structure, they applied the Delaunay triangulation and the Voronoi diagram to construct a routing graph, and proposed a three-stage algorithm to solve this problem, including (1) a network-flow-based algorithm for congestion estimation, (2) a maze-routing algorithm for PA nets on the routing graph, and (3) an MCMF-based routing algorithm for FA nets.

2) Existing InFO Routing: RDL routing for InFO packages have been addressed in the previous works [11], [12]. Lin *et al.* [11] addressed the first RDL routing problem for the multi-chip InFO package. They proposed a concentric-circle model to solve the layer assignment problem for PA nets and utilized the MCMF algorithm and ring-by-ring routing to obtain final results for FA nets. Lin *et al.* [12] considered an InFO *package-on-package (PoP) structure*, which contains frontside and back-

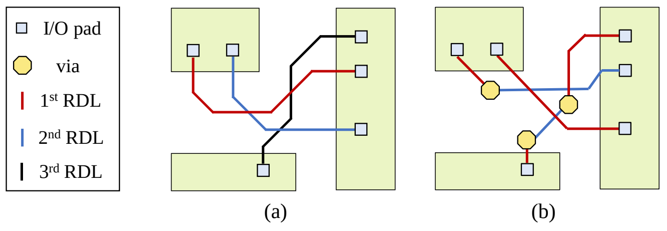

Fig. 2: A comparison between the previous work [11] and our algorithm. (a) The previous work [11] generates a routing result with three RDLs. Without flexible vias, each net can only be routed within one single layer. Therefore, the entangling three nets must be assigned to three different RDLs. (b) In contrast, our algorithm can generate a routing result with only two RDLs with appropriate via arrangement.

side RDLs connected by through-InFO-vias (TIVs). This work proposed an algorithm based on the directed acyclic graph which is transformed from the increasing subsequence to minimize the RDL number.

B. Emerging Package Considerations

Industrial InFO package designs contain much more complicated structures than those considered in the previous works. Previous works for InFO package routing [11], [12] and flip-chip routing [5] do not allow flexible RDL vias, and all pads are aligned with fixed vias punching through all RDLs for inter-layer connections. However, according to recent technical reports from TSMC [13], [2] and the IEEE heterogeneous integration roadmap [14], flexible RDL vias are now widely used in both wafer-level packages (such as InFO and CoWoS) and interposer-based 2.5D packages. Moreover, the via-induced signal integrity degradation issue mentioned in [5] can be alleviated by reliability-aware design techniques such as ground-signal shielding. As a result, via arrangement has become a crucial issue in RDL routing.

Existing methods forbiding flexible vias may suffer from restricted solution space on advanced package routing problems. Without flexible vias, each net must be routed within one single layer. Under this restriction, previous works adopted layer assignment for all pre-assigned nets to avoid wire crossings. Figure 2(a) gives an example of a routing result generated by the previous method [11]. In this example, three RDLs are needed to route all the three nets. Although layer assignment could minimize the required layer count without using vias, its solution quality is often suboptimal. Figure 2(b) shows a better solution with only two RDLs if flexible vias are available.

C. Our Contributions

We summarize the contributions of this paper as follows:

- We propose an RDL routing flow for InFO packages to maximize the routability and minimize the wirelength under limited RDLs. To the best of our knowledge, we are the *first* in the literature to formulate and solve the routing problem with pre-assigned pad pairs on a via-based multi-chip multi-layer InFO package with irregular pad structures. Our router is also the *first* to handle both flexible vias and irregular pad structures.
- We propose a weighted-MPSC-based algorithm to perform efficient single-layer concurrent routing by analyzing the multi-chip I/O structures in an early stage. By appropriately setting the weight of each chord, we can reduce the gap between the layer assignment and the detailed routing.
- We propose a novel *linear-programming (LP)-based* 3D layout optimization method for effective via arrangement. This method is applicable to all existing RDL topologies and guarantees to converge to a solution in a theoretically bounded number of iterations, given a legal initial layout.
- We propose a novel, effective octagonal tile model to estimate the routing resources for complicated designs with X-architecture wires and octagonal components.
- Experimental results show that our algorithm is flexible and effective for complex designs.

The remainder of the paper is organized as follows. Section II introduces the terminologies, the notations, the RDL structure, the design rules, and the problem formulation. Section III presents our proposed algorithm flow. Section IV reports and analyzes the experimental results. Finally, Section V concludes this paper.

II. Preliminaries

In this section, we first introduce the terminologies and the routing design rules considered in this paper, and then give our formulation of pre-assignment routing problem on via-based multi-chip multi-layer InFO packages.

A. Terminologies and Notations

The terminologies and notations are listed as follows:

- $Q = \{q_i \mid 1 \leq i \leq |Q|\}$ is the set of all I/O pads. I/O pads are rectangular pads attaching to the top RDL for chip-to-package contacts.
- $G = \{g_i \mid 1 \leq i \leq |G|\}$ is the set of bump pads. Bump pads are octagonal pads attaching to the bottom RDL for package-to-board contacts.
- $N = \{n_i \mid 1 \leq i \leq |N|\}$ is the set of pre-assigned nets. Each net indicates a pad pair that should be connected. For an inter-chip connection, two I/O pads are paired. For a chip-to-board connection, one I/O pad and one bump pad are paired.
- $O = \{o_i \mid 1 \leq i \leq |O|\}$ is the set of rectangular obstacles.
- $V_p = \{v_p^i \mid 1 \leq i \leq |V_p|\}$ is the set of pre-assigned (fixed) vias.
- $V_f = \{v_f^i \mid 1 \leq i \leq |V_f|\}$ is the set of flexible vias.
- s is the minimum spacing.
- s_w is the wire width.
- s_v is the via width. In this paper, a via is modeled as a regular octagon. Therefore, the via width is the width of its bounding box.
- Via layer (L_v): a via layer is an RDL which contains vias and obstacles.
- Wire layer (L_w): a wire layer is an RDL which contains metal wires and obstacles.

B. RDL Structure and Design Rules

In this work, we consider the irregular pad structure, i.e., all the I/O pads and bump pads can be placed at arbitrary positions as long as the minimum spacing rule is not violated. The design rules considered in this work are listed as follows:

- Minimum spacing rule: A minimum spacing is required between every pair of *components*, i.e., wire segments, vias, and obstacles, if the components belong to different nets.
- X-architecture wires: A wire segment in an RDL can only be routed in four kinds of orientations: vertical, horizontal, and 45/135-degree diagonal directions.
- Non-crossing constraint: Two nets cannot be routed across each other on an RDL.
- Routing-angle constraint: two connected wire segments can have a right-angle turn or a 135-degree turn. A 45-degree turn is not allowed for better manufacturability.

C. Problem Formulation

The pre-assignment routing problem on the via-based multi-chip multi-layer InFO package addressed in this paper can be formulated as follows:

- **The Pre-assignment Via-based Multi-layer Multi-chip RDL Routing Problem**: Given an RDL structure, a netlist of pre-assigned nets, and design rules, connect all the pre-assigned nets such that the routability is maximized, the total wirelength is minimized, and no design rules are violated.

III. Algorithms

In this section, we first provide an overview of our algorithm flow, and then detail the models and methods used in each stage. Figure 3 shows our proposed RDL routing flow which consists of the following five stages: (1) Preprocessing, (2) Weighted-MPSC-based Concurrent Routing, (3) Routing Graph Construction, (4) Sequential A*-search Routing, and (5) LP-based Layout Optimization.

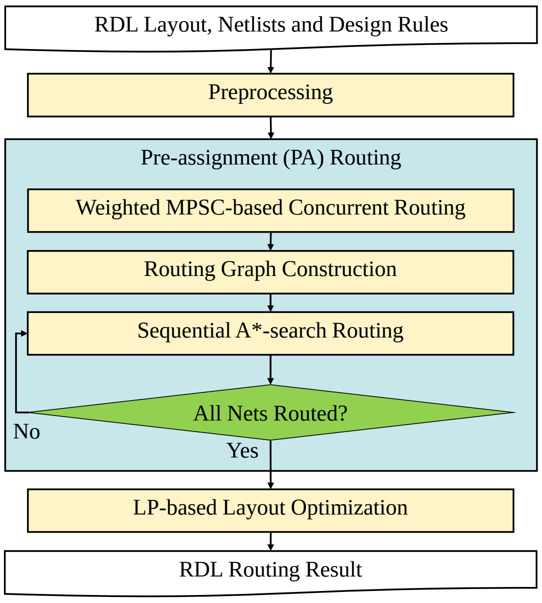
RDL Layout, Netlists and Design Rules
Preprocessing
Pre-assignment (PA) Routing
Weighted MPSC-based Concurrent Routing
Routing Graph Construction
Sequential A*-search Routing
All Nets Routed?
No
Yes
LP-based Layout Optimization
RDL Routing Result

Fig. 3: Our RDL routing flow.

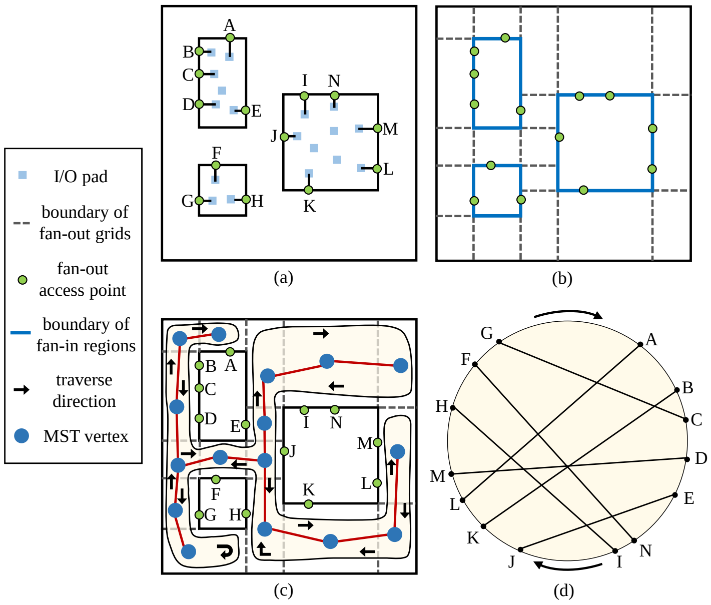

Fig. 4: Example of Preprocessing. (a) In Peripheral I/O Identification, the fan-out access points are decided. (b) In Fan-out Region Partitioning, the fan-out region is partitioned into fan-out grids. (c) In MST Construction, an MST is obtained to represent a routing topology among merged fan-out grids. (d) In Circular Model Construction, chords are constructed to represent net candidates.

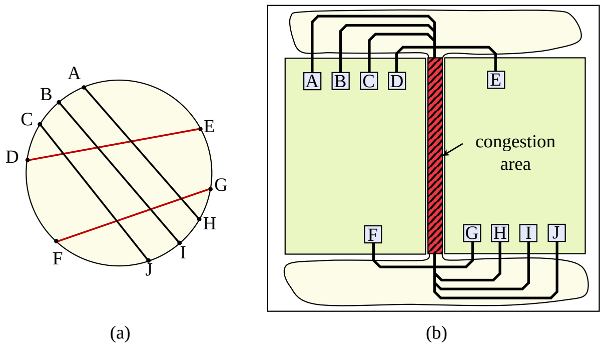

Fig. 5: An example of the routing congestion gap between the layer assignment and the detailed routing. (a) The result of Supowit's MPSC-based layer assignment would incur a routing failure in (b) because of the congested area.

In Preprocessing, we analyze the routing resources and the potential routing congestion in the *fan-out region*. We also identify the nets that could be concurrently routed in the fan-out region of a single RDL to reduce via usage. In Weighted-MPSC-based Concurrent Routing, we first perform Weighted-MPSC-based Layer Assignment to determine which nets should be routed in this stage. Then, we complete the detailed routing for these nets. After Weighted-MPSC-based Concurrent Routing, we construct an octagonal-tile-based routing graph from the current layout for the following Sequential A*-search Routing. In Sequential A*-search Routing, we complete the remaining connections by performing the A*-search algorithm on the routing graph. Finally, in LP-based Layout Optimization, we optimize the position of each component by solving an LP problem to minimize the total wirelength.

A. *Preprocessing*

To develop a better routing method, we find the regularity from the irregular pad structure by considering the fan-out region. Compared to the fan-in region, the fan-out region is not congested in the initial layout, thus is more suitable for concurrent routing. Therefore, we analyze the routing resources of the fan-out region and identify the net candidates to be routed in this region.

1) Peripheral I/O Identification: A net is identified as a candidate for the fan-out region concurrent routing if its corresponding I/O pads are located in the periphery of a fan-in region, because these I/O pads can directly be routed to the boundaries of the fan-in region, as shown in Figure 4(a). These corresponding boundary points are called *fan-out access points* for the fan-out region concurrent routing.

2) Fan-out Region Partitioning: In this step, we analyze the routing resources in the fan-out region. First, we apply Ohtsuki's partitioning method [15] to partition the fan-out region into rectangular grids. The method extends the boundaries of the fan-in regions until the extended lines meet other fan-in regions or the package boundary. Then, we apply the grid merging method proposed by Lee *et al.* [6] to avoid fragmented grids. The merged grids are called *fan-out grids*. Figure 4(b) gives the results of grid partitioning and merging of Figure 4(a).

3) Minimum Spanning Tree (MST) and Circular Model Construction: We extract a routing topology and estimate the routing congestion by pre-routing the net candidates in the fan-out region. After partitioning and merging, we first construct the fan-out grid graph. In the graph, each vertex represents a fan-out grid, and two vertices are connected by an edge if the corresponding grids are adjacent. For better efficiency, we construct an MST of the fan-out grid graph as a preferred pre-routing topology to guide the net candidates. The I/O pads of a net candidate are mapped to MST vertices, and the pre-routed path of a net candidate is the path on the MST. Figure 4(c) gives the MST to represent the fan-out region.

To determine the net candidates to be routed in a same RDL, we construct a circular model to represent the geometric relations between the fan-out region and the net candidates. We first construct a closed shape enclosing the MST, as shown in Figure 4(c). Then we obtain a sequence of the fan-out access points by traversing the boundary of the closed shape and reshaping the closed shape into a circular model. The sequence is assigned on the boundary of the circular model, and a net candidate is represented by a chord in this model. Figure 4(d) shows the final circular model.

B. *Weighted-MPSC-based Concurrent Routing*

In this stage, we use the circular model obtained from Preprocessing to perform concurrent routing. we first perform Weighted-MPSC-based Layer Assignment to assign as many nets as possible in the fan-out region of each RDL, and then we complete detailed routing to obtain an initial routing result for each pre-assigned net.

1) Weighted-MPSC-based Layer Assignment: To find a proper layer assignment, we extend the weighted MPSC algorithm based on Supowit's MPSC algorithm [16], which finds a maximum planar subset of nets in channel routing by transforming a routing channel and nets into a circular model and chords, respectively. Different from Supowit's work, our circular model is transformed from the whole fan-out region instead of a routing channel to obtain a global view. Moreover, we consider the capacity issue, which is highly related to the gap between the layer assignment and the detailed routing due to routing congestion. Figure 5 gives an example with five net candidates: (A,H), (B,I), (C,J), (D,E) and (F,G) in the fan-out region and its corresponding circular model. If we directly apply Supowit's MPSC algorithm on this model, the layer assignment result would be {(A,H), (B,I), (C,J)} with the maximum cardinality equal to three. However, only one of these three nets can be routed in detailed routing because these nets share a narrow channel, as shown in Figure 5(b). Therefore, the original MPSC algorithm may miss a better assignment {(D,E), (F,G)} for detailed routing.

To obtain a layer assignment result which is friendly to detailed routing, we modify the original MPSC algorithm by assigning a weight to each chord in the circular model. The weight of a chord is determined by the *overflow rate* and the *detour rate*. The detour rate $r_d(n)$ of a net (chord) n is the length of its pre-routed path divided by the distance between its two I/O pads, and the overflow rate $r_{ov}(e)$ of an MST edge e is defined by the following equation:

$$r_{ov}(e) = \begin{cases} 0, & \text{if } cap(e) \geq dem(e), \\ \frac{dem(e)}{cap(e)}, & \text{otherwise,} \end{cases} \tag{1}$$

where $cap(e)$ is the capacity of e representing the maximum number of nets simultaneously passing through the border of the fan-out grids, and $dem(e)$ is the demand of e representing the number of pre-routed net candidates that need to pass through e.

The weight of a chord u_n of a net candidate n is then calculated by

$$\begin{aligned} weight(u_n) = (&\alpha \cdot r_d(n) + \beta \cdot \log_\delta(\delta + f_{\max}(n)) \\ &+ \gamma \cdot \log_\delta(\delta + f_{avg}(n)))^{-1}, \end{aligned} \tag{2}$$

where α, β, γ and δ are user-defined parameters, $f_{max}(n)$ is the largest overflow rate of all MST edges in the pre-routed path of n, and $f_{avg}(n)$ is the average overflow rate of all MST edges in the pre-routed path of n.

A large $r_d(n)$ implies that the net would have a long detour if it is routed along the MST edges, a large $f_{max}(n)$ implies that the net would pass through a highly congested area, and a large $f_{avg}(n)$ implies that the whole path would be located in highly congested areas. All aforementioned cases imply that the net n should be prevented from being assigned in this stage, or a detailed routing failure might occur.

Finally, we perform the MPSC algorithm on the weighted circular model to obtain a layer assignment result with the maximum total weight.

2) Concurrent Detailed Routing: After layer assignment, we perform detailed routing for the assigned nets. According to the MST constructed in Section III-A3, we can obtain the routing topology of each net in the fan-out grids. Therefore, we can concurrently route all nets by performing pattern routing on the fan-out grids.

C. Routing Graph Construction

In this stage, we construct a routing graph for the remaining unrouted nets in the previous stage. In Global Cell Construction and Frame Partitioning, we partition an irregular layout based on routed nets and vias. In Octagon Tile and Routing Graph Construction, we transform the partition result into a routing graph. In Via Insertion, we insert vias into the layout to increase the routability and update the routing graph.

1) Global Cell Construction and Frame Partitioning: To construct a routing graph for the remaining unrouted nets, we first perform partitioning on the current layout. However, different from Fan-out Region Partitioning in Section III-A2, applying the partition method in the work [15] on the current layout would generate extremely fragmented grids because the number of components is much larger than the number of fan-in regions. Therefore, to avoid fragmented grid generation, a new partition method is needed. We first partition the layout into rectangular areas called *global cells*. The number of global cells can be adjusted according to the layout size and the number of components. We then apply the partition method in the work [15] to each global cell. As shown in Figure 6(a), we extend vertical and horizontal lines (the dotted lines in the figure) from corner points of vias and obstacles and endpoints of wire segments until they meet a boundary of a global cell or other components. With these extended lines, each global cell is further partitioned into smaller rectangular grids called *frames*.

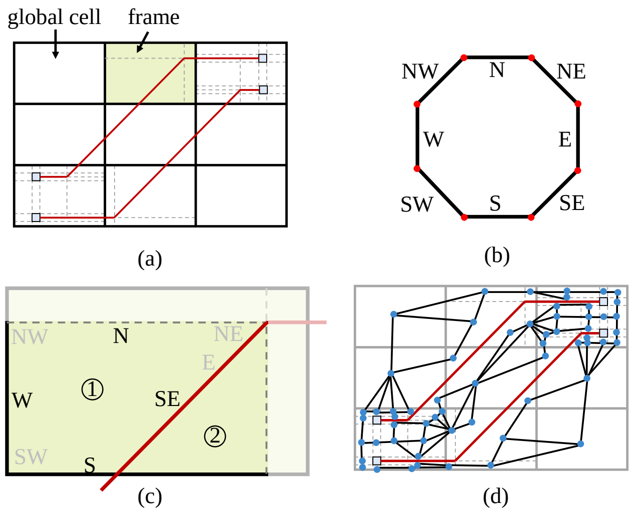

Fig. 6: Routing Graph Construction. (a) Global cells and frames generated by partitioning. (b) An octagonal tile with eight orientation-fixed boundary edges. (c) An octagonal tile represents a separated area inside a frame. (d) An octagonal-tile-based routing graph is generated to perform A*-search routing.

2) Octagon Tile and Routing Graph Construction: Figure 6(b) shows our octagonal tile model, which is an octagon enclosed by eight boundary edges of different orientations. The orientation of each boundary edge is fixed, but the length of each boundary edge is not restricted. Therefore, an octagonal tile model can represent any shape which is degraded from an octagon, such as triangles, rectangles, and even trapezoids with only 45-degree, 90-degree, and 135-degree interior angles.

Based on the partitioning result, we utilize our octagonal tile model to handle the diagonal structures in each frame. As shown in Figure 6(c), the frame is split into two separated areas by the red diagonal wire. Each area of the frame can be represented by an octagonal tile. For example, area (1) is regarded as an octagon with its northwest, southwest, northeast, and east boundary edges degrading to points. Figure 6(d) shows an example of Routing Graph Construction result. Each tile is represented by a vertex of the routing graph. Two vertices are connected by an edge if the corresponding tiles are adjacent and there is no blockage between these tiles.

3) Via Insertion: We perform via insertion in each global cell to generate inter-layer connections in the routing graph. For each global cell, we insert a via into the largest octagonal tile in that cell. This via would be projected through upper/lower layers until it reaches the top/bottom layer or a blockage. Finally, we update the routing graph by adding edges generated by via insertion and obtain the final multi-layer routing graph.

D. Sequential A*-search Routing

In this stage, we complete the routing based on the routing graph generated in the previous stage and generate an initial routing solution. We perform A*-search algorithm to sequentially route the nets. For each net, we first obtain a path consists of octagonal tiles by the A*-search algorithm. Then, we construct wire segments for a net in each octagonal tile according to the routing directions of the net inside the tile and the entering and leaving points of the net on the boundaries of the tile. After routing a net, the frames with the routed path are partitioned by the new wire segments, and the routing graph is updated accordingly.

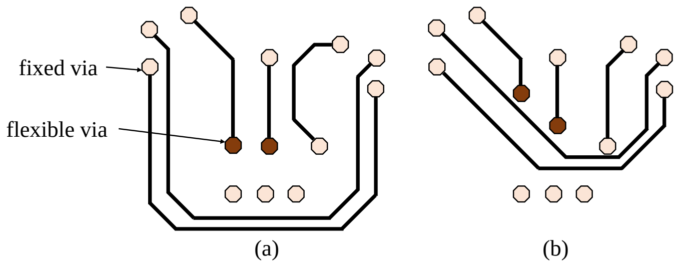

Fig. 7: An example of our LP-based Layout Optimization. (a) An initial routing solution. (b) An optimized layout with minimized wirelength generated by our LP-based Layout Optimization.

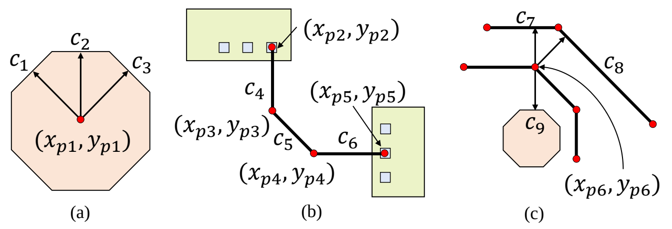

Fig. 8: Examples of LP variables and constraints: (a) Variables and fixed constraints for a via. (b) Variables for a route. (c) Variables and interactive constraints for a point in a route.

E. LP-based Layout Optimization

After obtaining an initial routing solution, we perform LP-based Layout Optimization to further minimize the wirelength. Figure 7 shows a scenario of our LP-based Layout Optimization. Our LP-based Layout Optimization contains four steps: (1) Layout Mapping, (2) Constraint Generation, (3) LP Problem Formulation, and (4) Iterative Solving. First, in Layout Mapping, we introduce a set of variables to represent the positions of each component, including obstacles, vias, and wires in the initial routing solution. Second, in Constraint Generation, we generate a set of linear constraints from the geometric relations in the initial routing solution. Third, in LP Problem Formulation, we set an objective function and formulate an LP problem from the generated constraints. Finally, in Iterative Solving, we solve the LP problem iteratively and obtain a final solution corresponding to an optimized layout.

1) Layout Mapping: First, we introduce necessary variables to represent an initial layout. There are three kinds of variables in our formulation: x variables, y variables, and c variables. The x and y variables represent *points*, and the c variables represent *segments*. Since every component can be represented by a set of points and segments, we can represent a layout by a set of x, y, and c variables.

- Point: the x- and y-positions of a point p in a 2D plane can be represented by a variable pair $(x_p,\ y_p)$.
- Line/segment: a line l in a 2D plane can be represented by a variable triplet $(a_l,\ b_l,\ c_l)$, because l in a 2D Cartesian coordinate system can be represented by an equation $a_l x + b_l y = c_l$. A segment can also be represented by the same equation form with additional endpoint conditions.

For segments, we only need to introduce the c variables because we would not modify the orientation of a segment and its connection topology during our optimization process. Therefore, the a and b variables and the endpoint conditions would be formed implicitly by linear constraints instead of variables.

We extract points and segments for each kind of components as follows:

- Obstacle: an obstacle is represented by a center point, four corner points, and four boundary segments.
- Via: a via is represented by a center point, eight corner points, and eight boundary segments.
- Route: a route is a routed net represented by a series of points and segments in a same layer, as shown in Figure 8(b).

2) Constraint Generation: To model the geometric relations and design rules into an LP formulation, we need to transform them into mathematical constraints. We classify the constraints for our LP problem into three types:

- Fixed constraint: A fixed constraint is to maintain a fixed geometric relation inside a component. For example, we should maintain a fixed distance (half of the via width) between the center point of a via and its boundary edges. As shown in Figure 8(a), fixed constraints between the center point of a via $(x_{p1},\ y_{p1})$ and its northwest, north, and northeast boundary edges can be written by

$$y_{p1} - x_{p1} = c_1 - \frac{\sqrt{2}}{2} \cdot s_v, \tag{3}$$

$$y_{p1} = c_2 - \frac{s_v}{2}, \tag{4}$$

$$x_{p1} + y_{p1} = c_3 - \frac{\sqrt{2}}{2} \cdot s_v. \tag{5}$$

- Route constraint: A route constraint is to maintain the connection relations inside a route. Take Figure 8(b) as an example. Two equality constraints are added for the point $(x_{p3},\ y_{p3})$ because it is the intersection of two lines ($x = c_4$ and $x + y = c_5$):

$$x_{p3} = c_4 \tag{6}$$

$$x_{p3} + y_{p3} = c_5 \tag{7}$$

- Interactive constraint: An interactive constraint is to avoid spacing rule violations between two components from different nets. In our method, we first extract interactive constraints for each point in all routes by finding the nearest blockage in each of the cardinal and intercardinal directions (N, NE, E, SE, S, SW, W, and NW) for each point. As the example shown in Figure 8(c), we extract three interactive constraints for the point $(x_{p6},\ y_{p6})$:

$$y_{p6} \leq c_7 - (s + \frac{s_w}{2}) \tag{8}$$

$$x_{p6} + y_{p6} \leq c_8 - \sqrt{2} \cdot (s + \frac{s_w}{2}) \tag{9}$$

$$y_{p6} \geq c_9 + (s + \frac{s_w}{2}) \tag{10}$$

3) LP Problem Formulation: Our objective for LP-based Layout Optimization is to minimize the wirelength. Therefore, the objective function of our LP problem is defined by:

$$\sum_{r \in R} \sum_{(p_i, p_{i+1}) \in r} d(p_i, p_{i+1}), \tag{11}$$

where R is the set of all planar routes, $r = \{p_1,\ p_2,\ ...,\ p_m\}$ is a planar route consisting of a series of m points, and $d(p_i, p_{i+1})$ is the length of a wire segment (p_i, p_{i+1}), which can be represented by the x and y variables in a linear form.

To sum up, our LP problem can be formulated as follow: *minimize the total wirelength, such that the fixed constraints, the route constraints, and the interactive constraints are satisfied.*

4) Iterative Solving: Since we do not specify all the geometric relations (interactive constraints) between each pair of components for efficiency consideration, a solution to the LP problem might be illegal due to wire crossing. In this case, we need to solve the LP problem for several iterations until we obtain a legal solution.

If the solution is illegal, we first identify all wire crossings in this solution. For each wire crossing, we consider the relative positions of the two wire segments in the initial layout and add corresponding interactive constraints. In the following iterations, we solve the LP problem along with those additional interactive constraints to avoid the wire crossings. This process guarantees to converge to a legal solution if the initial layout is legal, and the number of iterations needed for a converged legal solution is bounded by the number of variables, which is linear to the number of components in the initial layout. In practice, optimization for our largest benchmark takes no more than 50 iterations.

TABLE I: Benchmark statistics including the number of chips, the number of I/O pads ($|Q|$), the number of bump pads ($|G|$), the number of pre-assigned nets ($|N|$), the number of wire layers ($|L_w|$), and the number of via layers ($|L_v|$), and comparisons of the percentage of routability, total wirelength, and runtime for our work and Lin *et al.* [11].

Circuit	#Chips	$\|Q\|$	$\|G\|$	$\|N\|$	$\|L_w\|$	$\|L_v\|$	Routability (%)		Total Wirelength (μm)		Runtime	
							Lin-ext	Ours	Lin-ext	Ours	Lin-ext	Ours
dense1	2	44	324	22	3	4	77.3	100.0	> 83883	89299	3.92	33.7
dense2	3	92	784	46	3	4	91.3	100.0	> 247938	223636	25.46	42.7
dense3	5	160	308	80	5	6	97.5	100.0	> 258122	293548	19.87	101.9
dense4	6	222	684	111	5	6	70.3	99.1	> 413759	> 462008	4.62	129.8
dense5	9	522	1444	261	5	6	55.6	92.3	> 1060010	> 1581164	165.8	307.1
Comparisons							0.794	1.00	–	–	0.297	1.00

IV. Experimental Results

We implemented our proposed routing algorithm in the C++ programming language on an Intel Xeon 3.50GHz Linux workstation with 72G memory. In LP-based Layout Optimization, we used the Gurobi Optimizer [17] as the LP problem solver. We performed experiments on five RDL circuits with positions of I/O pads and pre-assigned net pairs. In our implementation, the parameters α, β, γ, and δ were set to 0.1, 1, 1, and 2 respectively, and the number of global cells was 900 (30×30).

To show the effectiveness and the completeness of our algorithm, we compared our algorithm with an integrated RDL routing flow, namely Lin-ext. Lin-ext adopted the concurrent routing method from the state-of-the-art InFO RDL routing work [11] and an A*-search-based sequential routing method to further improve its routability. Table I shows the benchmark statistics and the experimental results, where "#Chips" gives the number of chips, $|Q|$ the number of I/O pads, $|G|$ the number of bump pads, $|N|$ the number of pre-assigned nets, $|L_w|$ the number of wire layers, and $|L_v|$ the number of via layers. As shown in Table I, our proposed via-based RDL routing flow achieves a 20% routability improvement over Lin-ext with a 70% runtime overhead. Note that the wirelength reported in Table I counts only the routed nets. Therefore, only the wirelength of those testcases with 100% routability result could be correctly computed. For circuits dense1, dense2, and dense3, our work can achieve 100% routability while Lin-ext fails to complete the routing. Note that for the circuit dense2, we obtain a shorter wirelength than Lin-ext. For complicated circuits dense4 and dense5, both works fail to complete the routing. However, our work can achieve respective 99% and 92% routability, which is acceptable in the practical design flow.

The reasons why our proposed via-based RDL routing flow can achieve significantly better routability with a reasonable runtime overhead are analyzed as follows:

- Our Weighted-MPSC Layer Assignment considers the potential routing congestion in the whole fan-out region, which provides a global view for the layer assignment and thus increases the routability during concurrent routing. In contrast, Lin-ext does not consider potential routing congestion, and its concentric model for the layer assignment considers only local nets in the surrounding regions of one chip at a time, thus limiting the routability.
- LP-based Optimization effectively reduces the wirelength of the routed nets and releases the routing resources after concurrent routing. Therefore, the routability can be significantly increased during sequential routing.
- With our proposed Octagonal Tile Model and Via Insertion, we can generate a 3D routing graph for sequential routing to achieve a better routing resource integration. Instead, Lin-ext only uses the 2D routing graph for each RDL, leading to lower routability.

V. Conclusions

This paper is the first in the literature to consider flexible RDL vias and irregular pad structures for via-based multi-chip multi-layer InFO packages. We have presented an RDL routing algorithm which consists of three key steps: (1) Weighted-MPSC-based Concurrent Routing, (2) Routing Graph Construction with an octagonal tile model for the irregular layout partition, and (3) LP-based Layout Optimization for the wirelength minimization. The experimental results have demonstrated the effectiveness and the robustness of our algorithm.

References

[1] Daniel Yang and Stacy Wegner. Apple watch series 3 teardown. Accessed: 2019-11-25. [Online]. Available: https://www.techinsights.com/zh-tw/node/16538

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

[3] H.-C. Lee, Y.-W. Chang, and P.-W. Lee, "Recent research development in flip-chip routing," in *Proc. of IEEE/ACM ICCAD*, 2010, pp. 404–410.

[4] J.-W. Fang, I.-J. Lin, and Y.-W. Chang, "A network-flow-based RDL routing algorithmz for flip-chip design," *IEEE TCAD*, vol. 26, no. 8, pp. 1417–1429, 2007.

[5] J.-W. Fang and Y.-W. Chang, "Area-I/O flip-chip routing for chip-package co-design," in *Proc. of IEEE/ACM ICCAD*, 2008, pp. 518–522.

[6] Y.-K. Ho, H.-C. Lee, W. Lee, Y.-W. Chang, C.-F. Chang, I.-J. Lin, and C.-F. Shen, "Obstacle-avoiding free-assignment routing for flip-chip designs," *IEEE TCAD*, vol. 33, no. 2, pp. 224–236, 2014.

[7] T. Yan and M. D. F. Wong, "Correctly modeling the diagonal capacity in escape routing," *IEEE TCAD*, vol. 31, no. 2, pp. 285–293, 2012.

[8] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang, "An integer-linear-programming-based routing algorithm for flip-chip designs," *IEEE TCAD*, vol. 28, no. 1, pp. 98–110, 2009.

[9] 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, 2012.

[10] J.-W. Fang, M. D. F. Wong, and Y.-W. Chang, "Flip-chip routing with unified area-i/o pad assignments for package-board co-design," in *Proc. of ACM/IEEE DAC*, 2009, pp. 336–339.

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

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

[13] S.-P. Jeng, D.-J. Chen, H.-T. Lu, H.-W. Liu, C.-H. Lin, S.-T. Hung, and P.-Y. Chuang, "Integrated fan-out packages," U.S. Patent 10 347 574, Mar 28, 2019.

[14] IEEE Electronics Packaging Society. Heterogeneous integration roadmap. Accessed: 2019-11-25. [Online]. Available: https://eps.ieee.org/technology/heterogeneous-integration-roadmap.html

[15] T. Ohtsuki, "Gridless routers—new wire routing algorithms based on computational geometry," in *Proc. of IEEE ICCS*, 1985, pp. 802–809.

[16] K. J. Supowit, "Finding a maximum planar subset of a set of nets in a channel," *IEEE TCAD*, vol. 6, no. 1, pp. 93–94, 1987.

[17] Gurobi Optimization, LLC. Gurobi optimizer reference manual. Accessed: 2019-11-27. [Online]. Available: https://www.gurobi.com/