Redistribution Layer Routing for Integrated Fan-Out Wafer-Level Chip-Scale Packages [*]

Bo-Qiao Lin[1], Ting-Chou Lin[1], and Yao-Wen Chang[1,2]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
{bqlin, jameslin}@eda.ee.ntu.edu.tw; ywchang@ntu.edu.tw

ABSTRACT

The integrated fan-out (InFO) wafer-level chip-scale package (WLCSP) is an emerging packaging technology, which typically consists of multiple redistribution layers (RDLs) for signal redistributions among multiple chips. There is still no published work specifically on the RDL routing for the InFO WLCSP. Published RDL routing works consider different types of routing, namely free-assignment, pre-assignment, and unified-assignment routing, for a single chip. With the integration of multiple chips under the InFO WLCSP, however, previous works cannot achieve high efficiency or effectiveness with simple extensions. To remedy the deficiencies of poor interactions between chips and multiple RDLs, we formulate a new RDL routing problem for the InFO WLCSP and present the first work in the literature to handle the unified-assignment, multi-layer multi-chip RDL routing problem (without RDL vias), considering signal integrity, layer assignment, layer number minimization, and total wirelength minimization. We propose a *concentric-circle model* which models all the connections among one chip and all other chips. Based on this model, we assign the connections between chips to appropriate layers to avoid long detours. In addition, this model transforms the geometrical information of the pre-assignment connections among chips into a network-flow model to generate a routing prototype in a fan-out region not covered by any chip efficiently and effectively. Experimental results demonstrate the high quality and efficiency of our algorithm.

1. INTRODUCTION

With the increasing design complexity, many advanced packaging technologies have been proposed for system integration recently. Among these advanced packaging technologies, the integrated fan-out (InFO) wafer-level chip-scale package (WLCSP) [2, 11, 17] provides a promising alternative with a small form factor, a better signal-to-noise ratio, and improved thermal characteristics. A multi-chip InFO WLCSP uses micro-bumps and redistribution layers (RDLs) to connect neighboring chips [7]. As illustrated in Figure 1(a) for the InFO WLCSP, RDLs are the top metal layers of a chip, and multiple chips share the same RDLs in a multi-chip InFO WLCSP. In modern IC designs, I/O pads are usually placed along the boundaries of chips, called *peripheral I/Os*, and the RDLs are used to redistribute I/O pads to bump pads or connect I/O pads among different chips. Accordingly, an RDL router is needed to connect I/O pads to the I/O pads in other chips or the bump pads in the same chip or the fan-out region not covered by any chip. RDLs under a multi-chip InFO WLCSP are divided into two regions: (1) the fan-in region, which is the region directly underneath chips, and (2) the fan-out region, which is the region not belonging to the fan-in region. Therefore, I/O pads can be redistributed to the fan-out region outside the chips to increase the pin count. Moreover, passive devices such as inductors and capacitors can be formed over molding compound for lower substrate loss and higher electrical performance [11]. Molding compound also provides physical protection for the chips.

[*]This work was partially supported by AnaGlobe, MediaTek, RealTek, TSMC, MOST of Taiwan under Grant NSC 102-2221-E-002-235-MY3, NSC 102-2923-E-002-006-MY3, MOST 103-2221-E-002-259-MY3, MOST 103-2812-8-002-003, MOST 104-2221-E-002-132-MY3, and NTU under Grant NTU-ERP-105R8951.

ICCAD '16, November 07-10, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4466-1/16/11...$15.00
DOI: http://dx.doi.org/10.1145/2966986.2967070

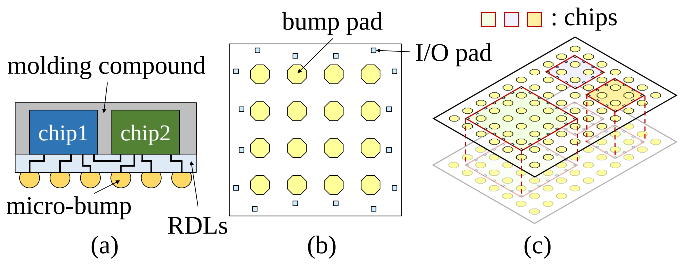


Figure 1: RDL structure comparisons between a traditional flip-chip and an InFO WLCSP. (a) The cross view of a multi-chip InFO WLCSP. (b) The RDL structure of a single chip without a fan-out region. (c) The RDL structure of a multi-chip InFO WLCSP with two RDLs.

1.1 Previous Works

Figure 1(b) shows a typical RDL structure for traditional flip-chip designs with peripheral I/Os, which contains only one chip with no fan-out region. Depending on the interactions among IC, packaging, and PCB designers, there are three types of traditional RDL routing problems: (1) the free-assignment routing problem, (2) the pre-assignment routing problem, and (3) the unified-assignment routing problem [9]. For the free-assignment routing problem, the assignment between I/O pads and bump pads are not predefined by designers. For the pre-assignment routing problem, in contrast, connections between I/O pads and bump pads are predefined by designers. For the unified-assignment routing problem, some connections between I/O pads and bump pads are predefined while the others are not. There is a series of related works on the RDL routing problems for traditional flip-chip designs. Fang *et al.* [5] addressed the free-assignment routing problem and proposed a network-flow-based algorithm to derive a global-routing topology. Liu *et al.* [12] also presented a network-flow-based method based on Delaunay triangulation and Voronoi diagram. The works [4, 10] addressed the pre-assignment routing problem. Fang *et al.* [4] adopted an integer-linear-programming-based method with the reduction techniques to prune redundant solutions to complete the global routing. Lin *et al.* [10] developed an efficient dynamic programming algorithm based on the concept of net sequence exchange. This router iteratively routed bump pads outwards, from the innermost bump pad ring to the outer bump pad rings. Fang *et al.* [6] addressed the unified-assignment routing problem. In this problem, an I/O pad can be placed at any location (called *area I/Os*). They proposed an algorithm consisting of three stages: (1) apply a network-flow-based method to route all the free-assignment connections to obtain the congestion information, (2) route the pre-assignment nets one by one considering the congestion information, and (3) modify the flow network according to the routed pre-assignment nets and complete the routing for the free-assignment nets.

Figure 1(c) shows the multi-layer RDL structure of a multi-chip InFO WLCSP. Different from the RDL structures of previous works, there are multiple chips and multiple layers. Besides, the types of connections are different. In addition to the original free-assignment connections, which connect I/O pads to bump pads to transmit signals into or out of a package, another type of connections under a multi-chip InFO WLCSP connects two I/O pads of different chips to directly transmit signals. In this addressed problem, the connections between I/O pads are predefined because the functions of I/O pads are typically predefined by IC designers. To solve this new problem, we could have extended the work [6] (modified unified-assignment) with the technique of handling multiple layers proposed in [3]; this extension modifies a flow network in [6] for multiple layers by duplicating the vertices and edges of the original network for each layer. However, as shown in Figures 2(a) and (b), this method may lead to long detours. The detours are caused by two reasons: (1) its congestion estimation does not consider pre-assignment nets, and (2) there exist net ordering problems for pre-assignment nets. To remedy the deficiencies of poor interactions between chips and multiple RDLs, we propose a new algorithm to achieve better solution quality, as shown in Figures 2(c) and (d).

1.2 Our Contributions

The main contributions of this paper are summarized as follows:

- We are the first in the literature to formulate the unified-assignment, multi-layer multi-chip RDL routing problem for the integrated fan-out (InFO) wafer-level chip-scale package (WLCSP) and propose the first algorithm to solve this problem.

- We propose a novel model (concentric-circle model) to reflect the geometrical information of pad connections. Based on this model, we can simultaneously consider all the pre-assignment nets of a chip and better solve the layer assignment problem. In particular, our proposed algorithm guarantees to find an optimal layer-assignment solution for a chip for some specific cases.

- We present a network-flow method to perform pad/layer assignment for free-assignment nets. Further, we integrate the geometrical information of pre-assignment nets into the flow network based on the proposed concentric-circle model. By considering pre-assignment nets with free-assignment nets, detours of pre-assignment nets caused by free-assignment nets can be effectively reduced.

- The experimental results show the effectiveness and efficiency of our algorithm. Specifically, our router achieves 100% routability for all the given testcases.

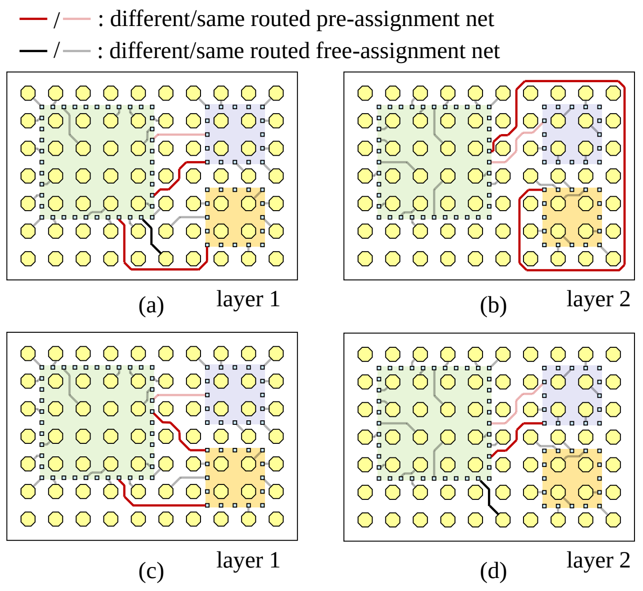

—/— : different/same routed pre-assignment net
—/— : different/same routed free-assignment net
(a)
layer 1
(b)
layer 2
(c)
layer 1
(d)
layer 2

Figure 2: Comparisons between the previous work [6] with extensions and our algorithm. (a)–(b) A two-layer routing result with detours, generated from the previous work [6] with the extension that modifies a flow network in [6] for multiple layers by duplicating the vertices and edges of the original network for each layer. The detours are caused by two reasons: (1) its congestion estimation does not consider pre-assignment nets, and (2) there exist net ordering problems for pre-assignment nets. (c)–(d) A better two-layer routing solution from our algorithm.

The remainder of this paper is organized as follows. Section 2 formulates the unified-assignment, multi-layer multi-chip RDL routing problem and gives the layout constraints. Section 3 details our proposed algorithm. Section 4 reports the experimental results. Finally, Section 5 concludes this paper.

2. PRELIMINARIES

In this section, we first define a new RDL routing problem and introduce the layout constraints for our problem.

2.1 Problem Formulation

The unified-assignment, multi-layer multi-chip RDL routing problem addressed in this paper is different from the previous works described in Section 1.1, with some additional issues considered. We first give the following notations and terminologies used throughout this paper:

- *Free-assignment I/O pad:* an I/O pad that has no predefined assignment. All the free-assignment I/O pads need to be connected to some bump pads.

- *Pre-assignment I/O pad:* an I/O pad that has a predefined assignment to a certain I/O pad.

- *Fan-in bump pad:* a bump pad located in the fan-in region.

- *Fan-out bump pad:* a bump pad located in the fan-out region.

- $B_o = \{b_1^o, b_2^o, ..., b_{k_o}^o\}$ is the set of fan-out bump pads, where k_o is the number of the fan-out bump pads.

- $B_{ij} = \{b_1^j, b_2^j, ..., b_{k_j}^j\}$ is the set of the bump pads in the region of chip c_j, where k_j is the number of the bump pads in the region of chip c_j.

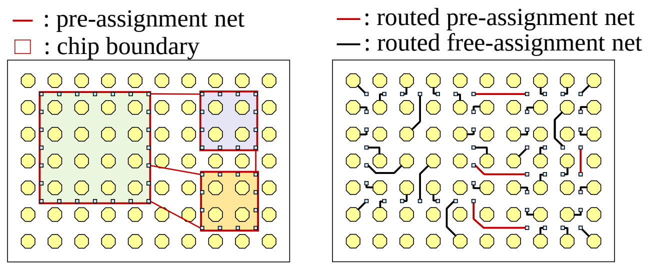


Figure 3: (a) The structure of the addressed RDL routing plane. Multiple chips are integrated into a single layout, and some pre-assignment nets need to be electrically connected in the final layout. Besides, each free-assignment I/O pad should be connected to a bump pad. (b) A single-layer routing result satisfying all the design rules corresponding to (a).

- $B_i = B_{i1} \cup B_{i2} \cup ...B_{id}$ is the set of all fan-in bump pads, where d is the number of chips.
- $B = B_o \cup B_i$ is the set of all bump pads.
- $Q_i = \{q_1^i, q_2^i, ..., q_{u_i}^i\}$ is the set of I/O pads that belongs to chip c_i, where u_i is the number of the I/O pads belonging to the chip c_i.
- $Q = Q_1 \cup Q_2 \cup ... \cup Q_d$ is the set of all I/O pads.
- $N = \{n_1, n_2, ..., n_m\}$ is the set of 2-pin pre-assignment nets, where m is the number of pre-assignment nets and n_i defines the connection between two I/O pads.

Figure 3(a) shows the structure of the considered RDL routing plane. Different from the previous works, multiple chips are integrated into a single layout, and some predefined I/O pads need to be electrically connected in the final solution. Figure 3(b) shows a single-layer routing result satisfying all the design rules corresponding to Figure 3(a). We formally define the problem as follows:

- **The Unified-Assignment Multi-Layer Multi-Chip RDL Routing Problem:** Given an RDL layout, pre-assignment netlists $n_i \in N$, and design rules, connect all pre-assignment nets $n_i \in N$ and connect all the other I/O pads to some bump pads $b \in B$ so that there is no net crossing and the total wirelength and the number of layers are minimized.

2.2 Layout Constraints of Our Problem

In this work, we consider the most popular peripheral I/O structure, i.e., all the I/O pads of a chip are placed along the chip boundaries. To the best of our knowledge, the RDL structure considered in this work is different from those of previous works, several specific layout constraints are required to reduce the design complexities or fabrication difficulties. We detail four major layout constraints of our problem:

- *Routing-region constraint:* Because the distribution of I/O pads on chip boundaries is very dense, detours/multiple crossings of a net on the chip boundaries are not allowed. As shown in Figure 4(a), the routed nets have some routed net segments in both the fan-in region (the region occupied by I/O pads is excluded) and the fan-out region, implying that some chip boundaries are crossed multiple times. In this work, we confine the routed nets with the routing-region constraint to avoid this undesired configuration.

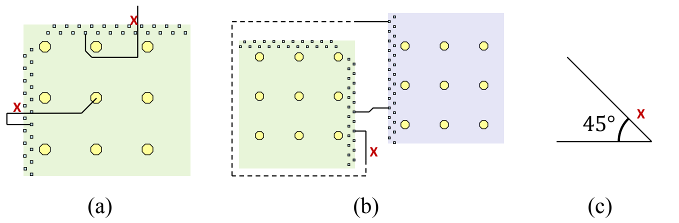


Figure 4: Three types of violations of the RDL layout constraints. (a) Violation of the routing-region constraint: both routed nets have some routed net segments in both the fan-in region (the region occupied by I/O pads is excluded) and the fan-out region. (b) Violation of the non-detour-around-boundary constraint: the routed net detouring around a large portion of the chip boundaries is not allowed. (c) Violation of the routing-angle constraint: the angle between two routed net segments should be 90° or 135°.

- *Non-detour-around-boundary constraint:* A net is not allowed to make a long detour around the chip; for example, we set the upper bound of 25% of the chip perimeter in our implementation. If a single net detours around a chip, significant routing resources might be occupied, and routing failures might occur eventually. As shown in Figure 4(b), the net that detours around the chip would occupy significant routing resources; besides, the signal skew problem might occur due to the resulting long route. As a result, for any two chips, there must exist continuous intervals on the chip boundaries without pre-assignment I/O pads connecting each other, or there is no routing solution which satisfies this constraint.
- *Routing-angle constraint:* A route can make a 90-degree (horizontally or vertically) or a 135-degree turn. However, a 45-degree turn is not allowed, as shown in Figure 4(c).
- *Non-crossing constraint:* Two nets cannot route across each other on the same layer. Furthermore, to maintain better signal integrity, each route is completed on a single layer, and vias are often not allowed (I/O pads and bump pads are punched through all RDLs) [3].

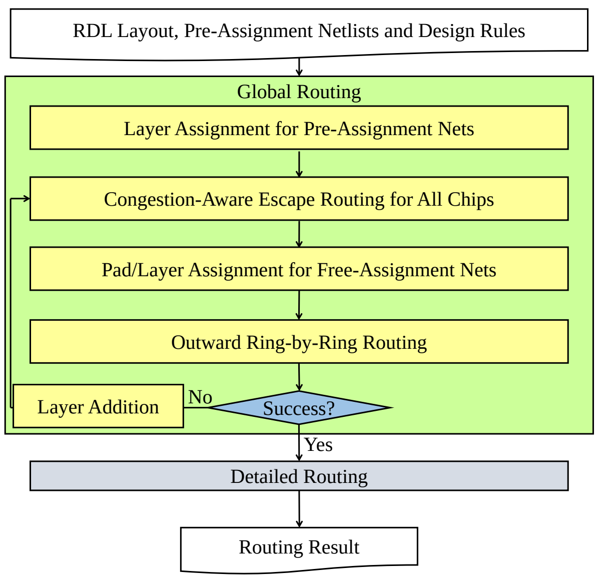
3. ALGORITHMS

In this section, we present our routing algorithm. We first give an overview of our algorithm, and then detail the methods used in each stage of the algorithm.

3.1 Algorithm Overview

As shown in Figure 5, our global routing algorithm consists of four stages: (1) layer assignment for pre-assignment nets, (2) congestion-aware escape routing for all chips, (3) pad/layer assignment for free-assignment nets, and (4) outward ring-by-ring routing.

In the first stage, we derive a *concentric-circle model* to represent the relative geometrical information of pre-assignment nets between one chip and all other chips, and then propose an efficient method to handle a layer-assignment problem for pre-assignment nets based on the derived model. With the layer-assignment information and congestion information, the second stage generates a routing prototype for fan-in bump pads by escape routing. The third stage constructs a flow network to assign free-assignment I/O pads to fan-out bump pads and escaped routes obtained in the second stage, and modifies the flow network with the concentric-circle model to consider pre-assignment nets. Finally, the fourth



RDL Layout, Pre-Assignment Netlists and Design Rules
Global Routing
Layer Assignment for Pre-Assignment Nets
Congestion-Aware Escape Routing for All Chips
Pad/Layer Assignment for Free-Assignment Nets
Outward Ring-by-Ring Routing
Layer Addition
No
Success?
Yes
Detailed Routing
Routing Result

Figure 5: Overview of our proposed routing algorithm.

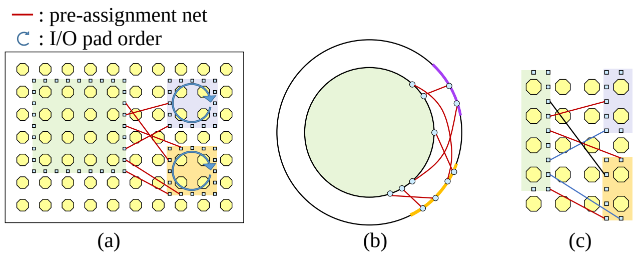


Figure 6: (a) An RDL layout with three chips and multiple pre-assignment nets. The blue arcs indicate the orders of corresponding nodes of I/O pads that appear on our concentric-circle model. (b) The corresponding concentric-circle model of (a). The inner circle represents the leftmost chip in (a), and the outer circle represents the neighboring geometry of the leftmost chip. The red arcs correspond to the pre-assignment nets connecting the nodes which represent the pre-assignment I/O pads. (c) A sample solution of the layer-assignment problem. The nets of the same color are assigned to the same layer.

stage applies an outward ring-by-ring technique to route the nets in the fan-out region. After the global routing, we apply the methods in [14] and [13] to obtain the detailed routing solution.

3.2 Layer Assignment for Pre-Assignment Nets

We start this section with a brief example. Given two pre-assignment nets as shown in Figure 4(b), if these two nets are routed on the same layer, a long detour will occur, which violates the non-detour-around-boundary constraint mentioned in Section 2.2. As a result, it is desired to avoid the aforementioned situation by assigning pre-assignment nets to appropriate layers.

To achieve a layer assignment with good quality, we propose a concentric-circle model and a method to handle the layer-assignment problem for all pre-assignment nets. Our method simultaneously assigns a number of pre-assignment nets for one chip and proceeds to the other chips on the same layer. We proceed to a new layer if there are pre-assignment nets yet to be assigned.

3.2.1 Concentric-Circle Model

Our concentric-circle model consists of two concentric circles. The inner circle represents the chip c_i, and the region outside the inner circle models the neighboring information of c_i. Besides the two circles, there are *nodes* and *bonds* in this model. We have the following definitions of a node and a bond.

DEFINITION 1. *A node in the concentric-circle model has the two attributes: a degree and its associated circle. The degree of a node in this model is the angle from the positive x-axis to the segment formed by the center of circles and the node. If a node is on the inner (outer) circle, we refer it as an inner (outer) node.*

DEFINITION 2. *A bond starts from an inner node to an outer one. We define a bond by two degrees and a direction to describe how the bond traverses from the inner node to the outer node. The two degrees of a bond are defined by the degrees of its inner and outer nodes. The direction of a bond is either clockwise or counter-clockwise.*

3.2.2 Construction of the Model for Layer Assignment for Pre-Assignment Nets

Each pre-assignment net belonging to the chip c_i can be modeled with a bond in the concentric-circle model. The inner node of the bond represents the I/O pad belonging to c_i. The outer node of the bond represents the other I/O pad. The degree of the inner node is determined by the angle from the positive x-axis to the segment formed by the center of c_i and the corresponding I/O pad. To determine the degree of the outer node, we first introduce the definition of a *degree interval*.

DEFINITION 3. *A degree interval is a continuous range of degrees. The degree interval of a bond is the range starting from its inner node to its outer one.*

Each chip to which c_i connects can be modeled with a degree interval on the outer circle. For example, the two right chips in Figure 6(a) are mapped to the non-overlapping degree intervals in Figure 6(b) on the outer circle. The center of the degree interval of a chip is decided by the angle from the positive x-axis to the segment formed by the centers of c_i and the chip. The degree intervals are chosen so that there is no overlap between them. The related pre-assignment I/O pads of each chip are mapped to the chip's degree interval starting from the I/O pad which is closest to the continuous interval on boundaries without pre-assignment I/O pads connecting to c_i (as mentioned in Section 2.2) clockwise. Given an inner node and an outer node, there are two possible bonds of opposite directions. We choose the bond with a smaller degree interval to represent the pre-assignment net to avoid traversing a large portion of the inner circle.

3.2.3 Algorithm Based on the Model

We observe that if two bonds intersect, assigning the corresponding pre-assignment nets to the same layer would cause an inevitable detour. As a result, it is desirable to find a maximum set of non-intersecting bonds and assign the set of nets to the same layer to minimize the number of layers. As shown in Figure 6, we first map the pre-assignment nets in Figure 6(a) to Figure 6(b) and perform the layer-assignment algorithm to obtain a sample solution as shown in Figure 6(c). In the following, we detail the method of finding a set of non-intersecting bonds.

To explain our method, we first define a *cutting bond*.

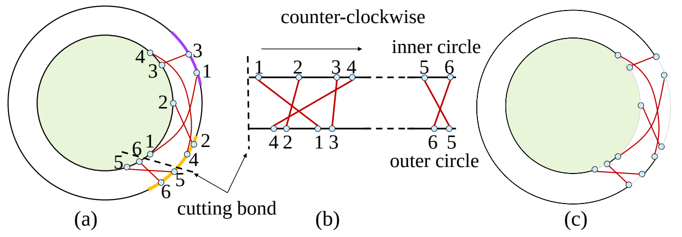


Figure 7: An example of our concentric-circle model in layer assignment for pre-assignment nets. (a) The same concentric-circle model as Figure 6 with a cutting bond. (b) The LCS problem converted from (a). The upper (lower) line corresponds to the inner (outer) circle in (a), and the nodes on the upper (lower) line follow the same order as on the inner (outer) circle, starting from the cutting bond counter-clockwise. (c) Candidate locations for a cutting bond on an inner (outer) circle are denoted by solid black arcs. If any two bonds intersect, some degree intervals on the two circles are excluded from consideration because no cutting bond will co-exist with an intersection.

Definition 4. *A cutting bond is a bond that does not intersect any bond on the concentric-circle model.*

As shown in Figure 7(a), if there is a cutting bond, we reduce the problem into a longest common subsequence (LCS) problem by transforming the cyclic structure of the model into a sequential structure, as shown in Figure 7(b). The nodes on the upper (lower) line follow the same order as on the inner (outer) circle, starting from the cutting bond counter-clockwise. If there are reversed numbers between the sequences of upper and inner lines, some bonds intersect. We apply the LCS algorithm in [15] on the two sequences to obtain a maximum set of non-intersecting bonds.

In order to find a cutting bond, we determine whether two bonds intersect in terms of degrees. If the directions of two bonds are the same, two bonds intersect if and only if the degree interval of one bond completely lies within the other one. If the directions of two bonds are different, two bonds intersect if and only if the degree intervals of the two bonds share some degrees. If any two bonds intersect, some degree intervals on the two circles are excluded from consideration because no cutting bond will co-exist with an intersection. As shown in Figure 7(c), the solid arcs on the inner (outer) circle are candidate degrees for a cutting bond on the inner (outer) circle. Therefore, we construct two interval trees to find all the intersections between all the bonds and identify the excluded degree intervals on the two circles. One of the interval trees represents the degree intervals of the clockwise bonds, and the other represents the degree intervals of counter-clockwise bonds. Then, we obtain candidate degree intervals on the two circles. We choose a degree on any inner candidate degree interval and form bonds with all outer candidates. Then, we use the interval trees to find a cutting bond. If there is no cutting bond, we find a bond intersecting the minimized number of the bonds on the concentric-circle model, remove these bonds and convert the problem into an LCS problem. The candidate inner (outer) degrees of the bond are any degree of degree intervals between any two adjacent nodes on the inner (outer) circle. Again, we use the two interval trees to find the bond intersecting the minimized number of the bonds on the model.

If some of the pre-assignment nets of the chip c_i have been assigned to the same layer, the original problem is decomposed into multiple LCS problems due to the corresponding bonds. Besides, if some previously assigned pre-assignment nets which do not connect to c_i, the concentric-circle model needs to be updated by trimming nodes on the corresponding degree intervals on the outer circle.

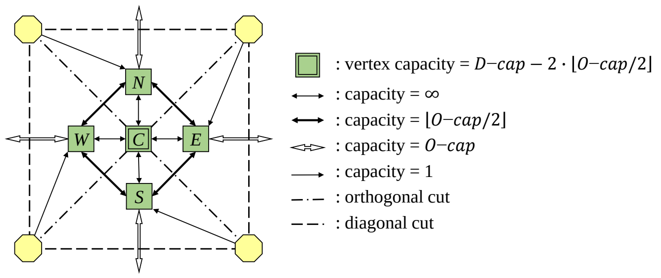

N
W
C
E
S
: vertex capacity = $D\text{-}cap - 2 \cdot \lfloor O\text{-}cap/2 \rfloor$
: capacity = ∞
: capacity = $\lfloor O\text{-}cap/2 \rfloor$
: capacity = $O\text{-}cap$
: capacity = 1
: orthogonal cut
: diagonal cut

Figure 8: The network flow model inside a tile with four bump pads proposed by [16].

We have the following theorem for the time complexity of the method.

Theorem 1. *If there is a cutting bond, our method finds a maximum set of non-intersecting bonds in $O(m_i \lg m_i + I_1 + I_2)$ time; otherwise, the method obtains a maximized set of non-intersecting bonds in $O(m_i^2 \lg m_i + I_3)$ time, where m_i is the number of the pre-assignment nets of the chip c_i, I_1 is the number of the intersections between bonds of pre-assignment nets, I_2 is the number of the intersections discovered in the checking process for finding a cutting bond, and I_3 is the number of the intersections discovered in the process for finding a bond intersecting the minimized number of the bonds.*

Proof 1. *We can obtain the concentric-circle model in $O(m_i)$ time, where m_i is the number of the pre-assignment nets of the chip c_i. The two interval trees can be obtained in $O(m_i \lg m_i)$ time. The time complexity of querying all the intersections between all the bonds is $O(m_i \lg m_i + I_1)$. Because the number of candidate degree intervals on the outer circle is $O(m_i)$, we can obtain a cutting bond (if it exists) in $O(m_i \lg m_i + I_2)$ time. Finally, we apply the LCS algorithm proposed in [15] whose time complexity is $O(m_i \lg m_i)$. Note that the numbers on the upper (lower) line are all different, the time complexity is thus reduced from $O(m_i^2)$ to $O(m_i \lg m_i)$. If there is no cutting bond, we try $O(m_i^2)$ choices of bonds to find out the bond intersecting the minimized number of the bonds on the model in $O(m_i^2 \lg m_i + I_3)$ time.*

3.3 Congestion-Aware Escape Routing for All Chips

After finishing the layer assignment for pre-assignment nets, we perform escape routing for all chips considering congestion information to finish the routing in the fan-in region. The escape routing problem has been extensively studied in recent years, which is to route specified bump pads in a bump pad array to the boundaries of the array [16].

Because multiple chips share the same RDLs, we have to consider the routing resource allocation in the fan-out region during escape routing. Based on the tile model proposed in [16], we further make some modifications to perform escape routing while minimizing potential routing congestion. Here, we make a brief review on the tile model. A tile consists of four adjacent bump pads. See Figure 8 for the flow network model of a tile. *O-cap* denotes the maximum number of nets which can pass through one side of the tile simultaneously, and *D-cap* is the maximum number of nets which can pass through the diagonal of the tile simultaneously. There are two super vertices, the source vertex and the sink vertex. There are directed edges from the source vertex to the bump pads which need to escape to the array boundaries. All

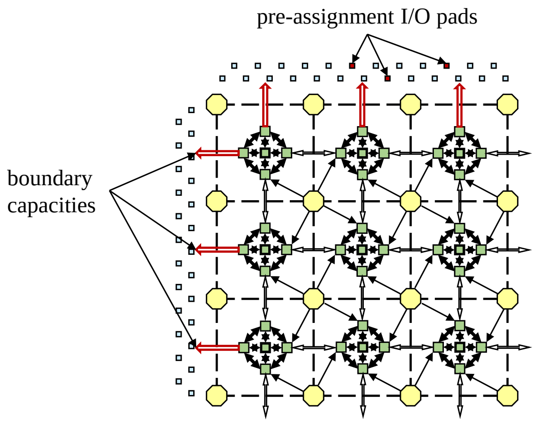


Figure 9: An example of the flow network of one chip. By adjusting the boundary capacities of the flow network according to (1) the resulting cost of a bipartite matching and (2) the locations of pre-assignment I/O pads, we can achieve a good distribution of nets after escape routing considering the congestion of the whole layout.

edges, the red arrows in Figure 9, from the boundary tiles to the outside of the array are connected to the sink vertex of a flow network. The inter-tile edges are assigned unit costs, while all the other edges are assigned zero costs. To handle multiple layers, we duplicate some vertices and edges for each layer without duplicating the vertices for bump pads (similar to the technique in [3]). After the flow network construction, we apply the minimum-cost maximum-flow (MCMF) algorithm [1] to obtain escaped routes while minimizing the total wirelength.

We adjust the capacities of the edges connected to the sink vertex (the red arrows in Figure 9) to control every maximum number of routes passing through all intervals of the array boundaries. To decide the value of its capacity, we calculate the cost (not the cost in the flow network) for each edge connecting to the sink vertex. The cost contains the bipartite cost C_B and the pre-assignment cost C_P.

$$cost = \alpha C_B + \beta C_P, \tag{1}$$

where α and β are user-defined parameters. C_P is equal to the number of neighboring pre-assignment I/O pads on the same layer. For C_B, we construct a bipartite graph $G_B = (V_B, E_B)$, where V_B is a set of vertices corresponding to all the free-assignment I/O pads and bump pads, and E_B is a set of edges. For a free-assignment I/O pad and a fan-out bump pad, we construct an edge between their corresponding vertices. For a free-assignment I/O pad q_j^i and a fan-in bump b_k^i (both pads belong to the chip c_i), we construct an edge between their corresponding vertices. Each edge has an associated weight which equals the Euclidian distance between the corresponding pads. Then, we perform the minimum weight maximum cardinality bipartite matching algorithm [8]. For each edge connecting to the sink vertex, the corresponding bipartite cost is the sum of the matched edge weights of the neighboring free-assignment I/O pads. The capacity of an edge connecting to the sink vertex is proportional to the cost and less than *O-cap*.

We have the following theorem for the time complexity of the method.

THEOREM 2. *The time complexity of this method is* $O((|B| + |Q|)^3 + L^2 \sum_j |B_{ij}|^3)$*, where B is the set of all the bump pads, Q is the set of all the I/O pads, L is the current number of layers, and* B_{ij} *is the set of the bump pads belonging to the chip* c_j.

PROOF 2. *The time complexity of the minimum weight maximum cardinality bipartite matching algorithm is* $O((|B| + |Q|)^3)$ *because the number of vertices in* G_B *is* $O(|B| + |Q|)$*. The time complexity of the MCMF algorithm for a chip* c_j *is* $O(L^2 |B_{ij}|^3)$ *because the maximum flow of the network for* c_j *is* $O(|B_{ij}|)$*, and the number of vertices is* $O(L |B_{ij}|)$.

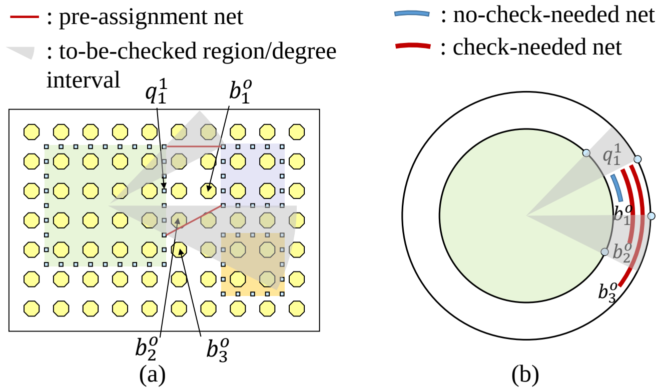


Figure 10: An example of our concentric-circle model in the stage of pad/layer assignment for free-assignment nets. (a) A sample layout with two pre-assignment nets. q_1^1 is the I/O pad of consideration, and b_1^o, b_2^o, and b_3^o are the candidate bump pads to be connected to. The to-be-checked regions are constructed from the perspective of the leftmost chip based on the pre-assignment nets connecting to it. (b) The corresponding concentric-circle model of (a). The inner circle represents the leftmost chip in (a). The red and blue arcs are the degree intervals derived from the layout. Starting from q_1^1, the red ones are the ones that need to be checked whether a potential detour may occur because they overlap with the bottom to-be-checked region, whereas the blue one needs not be checked.

3.4 Pad/Layer Assignment for Free-Assignment Nets

After the escape routing stage, we construct a flow network and apply the minimum-cost maximum-flow (MCMF) algorithm [1] to assign all free-assignment I/O pads to the fan-in region (the escaped routes and the outermost bump pads of bump pad arrays inside the chips) and the fan-out region (the fan-out bump pads). We refer to the outermost bump pads of bump pad arrays inside the chips as *boundary bump pads*. Note that at this stage, we have layout information of the finished routes in the fan-in region and the layer-assignment results of pre-assignment nets (actual routes are yet to be determined). To fully integrate all the layout information into our network-flow model, we remove some edges while considering the pre-assignment nets.

First, we introduce the construction of the flow network. Then, we modify the flow network considering the pre-assignment nets. Every free-assignment I/O pad and every escaped route correspond to a vertex in the flow network. For each fan-out and boundary bump pad, L duplicated vertices are created to represent the same-location bump pad on each layer, where L is the current number of layers. To ensure that each fan-out and boundary bump pad is assigned only on one layer, another vertex as a super vertex is created for each fan-out and boundary bump pad. In addition, two super vertices, which are the source vertex and the sink vertex, are created in the flow network. There are six types of directed edges:

- Edges from the vertex for a free-assignment I/O pad to the L duplicated vertices for a fan-out bump pad or a neighboring boundary bump pad.

- Edge from the vertex for a free-assignment I/O pad to the vertex for a neighboring escaped route.
- Edges from the L duplicated vertices for a fan-out bump pad or boundary bump pad to the super vertex for the same bump pad.
- Edge from the source vertex to the vertex for a free-assignment I/O pad.
- Edge from the super vertex for a fan-out bump pad or boundary bump pad to the sink vertex.
- Edge from the vertex for an escaped route to the sink vertex.

Every edge has a unit capacity. The edge cost of the first type is the Euclidian distance between the two pads. The edge cost of the second type is the wirelength of the escaped route plus the Euclidian distance between the I/O pad and the bump pad array boundary location of the escaped route. All the other edges have zero costs.

To consider the pre-assignment nets, we need to remove some edges in the flow network to avoid potential detours or routing failures. As shown in Figure 10(a), assigning the I/O pad q_1^1 to the bump pad b_3^o would lead to an inevitable detour, due to the bottom pre-assignment net. We use the concentric-circle model to identify edges which might lead to detours and remove them. For each chip on each layer, we first construct a concentric-circle model with bonds of the pre-assignment nets of the chip on the layer. Taking Figures 10(a) and (b) as an example, the gray regions are *to-be-checked regions (degree intervals)* in which, if other nets exist, some detours might occur. Note that the method to decide the degree of an outer node is different from the one in Section 3.2. To map a pre-assignment or a free-assignment net to a bond, the degree of its outer node is decided by the angle from the positive x-axis to the segment formed by the center of the chip to the I/O pad or the bump pad. After the construction of the concentric-circle model for each chip on each layer, all the to-be-checked degree intervals are formed based on the pre-assignment nets. We then construct an interval tree to store all the to-be-checked degree intervals for each chip on each layer. We use interval trees to decide whether edges of the overall flow network may need to be removed to avoid potential detours. Finally, we check the results obtained by the interval trees to actually remove them by the physical locations of the pads. As shown in Figure 10(b), the edge from q_1^1 to b_2^o and the edge from q_1^1 to b_3^o may need to be removed (we exclude the edge from q_1^1 to b_1^o based on the interval tree). According to Figure 10(a), the edge from q_1^1 to b_3^o should actually be removed.

Because we use the Euclidian distance as the cost metric and consider the pre-assignment nets, very few crossings, if not zero, will be introduced for practical applications. In addition, we can distribute I/O pads to all the layers evenly by adjusting the edge costs. There is an additional cost on the edges from an I/O pad to the vertices on some layers. Because there is a minimum distance between an I/O pad and a bump pad, we can choose the additional cost small enough which would not increase the number of crossings. Hence, the result of the MCMF algorithm often gives a good assignment topology which facilitates the following stages.

We have the following theorem for the time complexity of the method.

Theorem 3. *The overall time complexity of this method is $O(|Q|\,(L\,|B| + |Q|)^2)$, where Q is the set of all the I/O pads, L is the current number of layers, and B is the set of all the bump pads.*

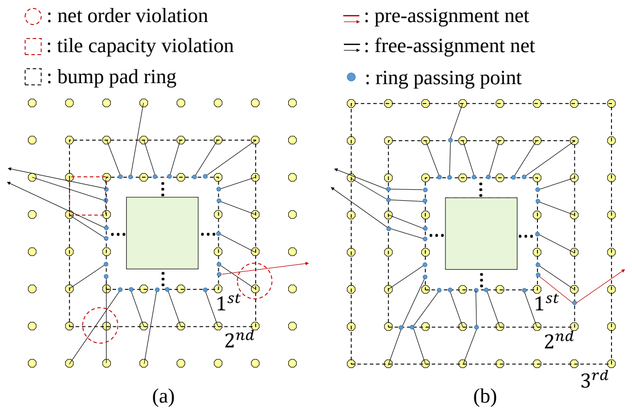


Figure 11: An example for our outward ring-by-ring routing algorithm. (a) The global routing result with two types of violations after finishing the first ring. All the I/O pads are assigned to correct ring passing points on the first ring. The ring passing points on the first ring are connected to the targets. (b) The global routing result after finishing the second ring. All the violations in (a) are corrected by reassigning some I/O pads to bump pads or detouring the pre-assignment net.

Proof 3. *We can construct the original flow network in $O(|B|\,|Q|)$ time. The time complexity of removing undesired edges is $O(|B|\,|Q|\,lg\,|N| + I_4)$, where $I_4 = O(|B|\,|Q|\,|N|)$ is the number of intersections between the degree intervals of bonds of free-assignment nets and pre-assignment nets. The time complexity of the MCMF algorithm is $O(|Q|\,(L\,|B| + |Q|)^2)$ because the maximum flow is $O(|Q|)$ and the number of vertices is $O(L\,|B| + |Q|)$. Hence, the time complexity of the method is dominated by the MCMF algorithm.*

3.5 Outward Ring-by-Ring Routing

Based on the MCMF result, all connections of I/O pads are determined. In this stage, we will finish the routing topology for each net in the fan-out region, based on a ring-by-ring, chip-by-chip scheme. Starting from the I/O pads, for each ring formed by the bump pads, we will assign correct ring passing points for all I/O connections in the fan-out region. To correctly assign the ring passing points, we have to fix two types of violations between two adjacent rings: (1) the net order violation and (2) the tile capacity violation. Because net crossings are not allowed, the net order on any two adjacent rings should match. Besides, we have to check if a given set of currently assigned net segments can pass through a tile simultaneously.

First, we can fix net order violations by either reassigning some I/O pads to bump pads, detouring the pre-assignment net, or reassigning some I/O pads to escaped routes or boundary bump pads in the fan-in region, as shown in Figure 11. Pre-assignment nets break two adjacent rings into many segments. The net orders of two corresponding segments on the inner and outer rings should be the same. If all order violations are free-assignment nets, we can reassign them to correct the order. If a net order violation involved a pre-assignment net, we have to detour the pre-assignment net or reassign free-assignment I/O pads to the fan-in region with unassigned escaped routes or boundary bump pads.

After correcting all net order violations, we check capacities of tiles by the tile routability analyzer proposed in [10]. The analyzer can check if a given set of currently assigned net segments can pass through a tile simultaneously in constant time. If a tile capacity violation occurs, we cluster the tile with an adjacent tile to push

Table 1: Benchmark circuit statistics and comparisons of RDL routing results ("N/A": incomplete routing results).

Circuits	#Chips	$\|Q\|$	$\|B\|$	$\|N\|$	#Layers		Routability (%)		Total Wirelength (μm)		Runtime (sec)	
					FA-maze	Ours	FA-maze	Ours	FA-maze	Ours	FA-maze	Ours
info1	3	672	672	44	3	2	95.4	100.0	N/A	142102	52	3
info2	3	1275	1364	89	3	2	93.5	100.0	N/A	514216	759	36
info3	3	1398	1600	107	3	2	92.4	100.0	N/A	611851	1117	41
info4	2	1672	1920	156	3	2	90.8	100.0	N/A	851808	1723	85
info5	4	3670	3720	376	3	3	89.3	100.0	N/A	1968153	15512	859
Comparisons					-	-	0.92	1.00	-	-	20.80	1.00

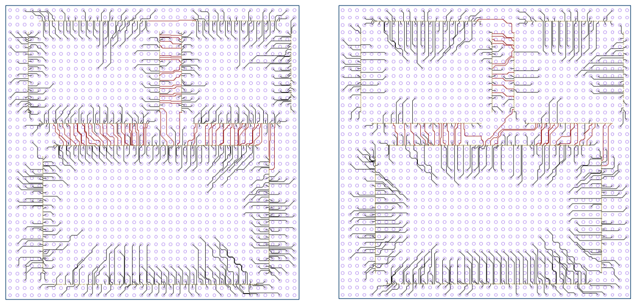


Figure 12: The RDL routing solution of info3. The red lines are the pre-assignment nets, and the black lines are the free-assignment ones. (a) First layer of the routing solution. (b) Second layer of the routing solution.

some net segments into the adjacent tile, as shown in Figure 11. It may need to cluster more adjacent tiles to fix a tile capacity violation. We can also reassign some I/O pads to escaped routes or boundary bump pads in the fan-in region.

4. EXPERIMENTAL RESULTS

We implemented our algorithm in the C++ programming language. All experiments were performed on an Intel Xeon 2.97GHz Linux workstation with 48 GB memory. The benchmark circuits are listed in Table 1, where "#Chips", "$|Q|$", "$|B|$', and "$|N|$" denote the numbers of chips, I/O pads, bump pads, and pre-assignment nets, respectively.

We compared our algorithm with a heuristic, FA-maze. FA-maze first routes all the free-assignment nets using the network-flow-based method proposed in [5] with the tile model proposed in [16]. We extend the work [5] to handle multiple layers with the technique presented in [3]. Then, the pre-assignment nets are routed sequentially by maze routing. When FA-maze cannot route all the nets using the current number of layers, we would add one more layer and perform FA-maze again. We set an upper bound on the maximum number of layers to three. The experimental results are shown in Table 1. The number of layers, the routability, the total wirelength, and the runtime are reported. From the results, our algorithm achieves 100% routibility for each circuit, while FA-maze fails to obtain a routing solution with more layers than ours. Further, our algorithm runs 20.8X faster than FA-maze. The results show that our algorithm is effective and efficient for the RDL routing for the InFO WLCSP.

5. CONCLUSIONS

This paper is the first in the literature to formulate the unified-assignment, multi-layer multi-chip RDL routing problem for the InFO WLCSP and proposes the first algorithm to solve this problem. To handle the layer-assignment problem for pre-assignment nets, we have proposed the concentric-circle model to model all the nets between one chip and all the other chips. Based on this model, we have assigned the nets to appropriate layers to avoid long detours. In addition, we have used this model to integrate the geometrical information of the pre-assignment nets between chips into a network-flow model, and the result can facilitate the following outward ring-by-ring stage and the detailed routing stage. Experimental results demonstrate the high quality and efficiency of our algorithm.

6. REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993.

[2] H. Chen, H.-C. Lin, C.-N. Peng, and M.-J. Wang. Wafer level chip scale package copper pillar probing. In *Proc. of ITC*, pages 1–6, October 2014.

[3] J.-W. Fang and Y.-W. Chang. Area-I/O flip-chip routing for chip-package co-design considering signal skews. *IEEE TCAD*, 29(5):711–721, May 2010.

[4] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang. An integer-linear-programming-based routing algorithm for flip-chip designs. *IEEE TCAD*, 28(1):98–110, January 2009.

[5] J.-W. Fang, I.-J. Lin, Y.-W. Chang, and J.-H. Wang. A network-flow-based RDL routing algorithmz for flip-chip design. *IEEE TCAD*, 26(8):1417–1429, August 2007.

[6] 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 DAC*, pages 336–339, July 2009.

[7] Y.-C. Huang, B.-Y. Lin, C.-W. Wu, M. Lee, H. Chen, H.-C. Lin, C.-N. Peng, and M.-J. Wang. Efficient probing schemes for fine-pitch pads of info wafer-level chip-scale package. In *Proc. of DAC*, pages 58:1–58:6, June 2016.

[8] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[9] H.-C. Lee, Y.-W. Chang, and P.-W. Lee. Recent research development in flip-chip routing. In *Proc. of ICCAD*, pages 404–410, November 2010.

[10] 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*, 31(6):878–889, June 2012.

[11] C. C. Liu, S.-M. Chen, F.-W. Kuo, H.-N. Chen, E.-H. Yeh, C.-C. Hsieh, L.-H. Huang, M.-Y. Chiu, J. Yeh, T.-S. Lin, T.-J. Yeh, S.-Y. Hou, J.-P. Hung, J.-C. Lin, C.-P. Jou, C.-T. Wang, S.-P. Jeng, and D. C. H. Yu. High-performance integrated fan-out wafer level packaging (InFO-WLP): Technology and system integration. In *Proc. of IEDM*, pages 14.1.1–14.1.4, December 2012.

[12] X. Liu, Y. Zhang, G. K. Yeap, C. Chu, J. Sun, and X. Zeng. Global routing and track assignment for flip-chip designs. In *Proc. of DAC*, pages 90–93, June 2010.

[13] D. Staepelaere, J. Jue, T. Dayan, and W. W.-M. Dai. Surf: Rubberband routing system for multichip modules. *IEEE Design & Test of Computers*, 10(4):18–26, December 1993.

[14] D. J. Staepelaere. *Geometric Transformations for a Rubber-Band Sketch.* University of California Santa Cruz, 1992.

[15] T. Szymanski. A special case of the maximal common subsequence problem. Technical report, Technical Report TR-170, Computer Science Laboratory, Princeton University, 1975.

[16] T. Yan and M. D. F. Wong. Correctly modeling the diagonal capacity in escape routing. *IEEE TCAD*, 31(2):285–293, February 2012.

[17] D. Yu. A new integration technology platform: Integrated fan-out wafer-level-packaging for mobile applications. In *Proc. of the Symposium on VLSI Technology*, pages T46–T47, June 2015.