Redistribution Layer Routing for Wafer-Level Integrated Fan-Out Package-on-Packages

Ting-Chou Lin[1], Chia-Chih Chi[2] 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
jameslin@eda.ee.ntu.edu.tw; b02901136@ntu.edu.tw; ywchang@ntu.edu.tw

***Abstract*—The wafer-level integrated fan-out (InFO) package-on-package (PoP) is a promising 3D packaging technology, which usually consists of a bottom package with the InFO technique, and a top package stacked on the bottom package. Different from the traditional PoPs, there are frontside and backside redistribution layers (RDLs) in the InFO PoP for signal redistributions. To the best of our knowledge, there is still no previous work specifically tackling the RDL routing for the InFO PoP. Previous works on RDL routing mainly deal with the following three types of routing: the free-assignment, pre-assignment, and unified-assignment routing for single or multiple chips. In this paper, a new RDL routing problem for the InFO PoP is formulated. To remedy the deficiencies of lacking the interactions between frontside and backside RDLs, we present the first work in the literature to handle the unified-assignment multi-layer multi-package RDL routing problem (without RDL vias), considering layer assignment, layer number minimization, and total wirelength minimization. We propose an algorithm based on extracting increasing subsequences (IS), which transforms a routing sequence into two directed acyclic graphs (DAGs), namely, *IS-DAG* and *Constraint-DAG*. By minimizing the number of vertices on the longest path on the Constraint-DAG, we implicitly minimize the layer number. Furthermore, we perform backtracking on the IS-DAG to efficiently assign the connections to appropriate layers to avoid long detours. Experimental results show that our router can achieve 100% routablility for all given test cases, while the previous works with extensions fail all test cases even with more frontside RDLs.**

I. Introduction

Recently, many advanced 3D packaging techniques have been proposed to provide multi-functional integration considering high pin counts, sub-system performance, and power efficiency. The wafer-level integrated fan-out (InFO) package-on-package (PoP) [11, 14, 16] is one of the 3D packaging techniques, which has better thermal performance, smaller form factor, lower cost, and improved integration flexibility. Figure 1(a) shows a cross-sectional view of a single-chip InFO PoP. Basically, a single-chip InFO PoP consists of a bottom package with the InFO technique, and a top package stacked on the bottom package. Mostly, the bottom package contains a logic chip, and the top package contains memory or other devices. In addition, the two packages are connected by through-InFO vias (TIV's) [14, 16], which are placed peripherally inner the boundary of the bottom package. Different from conventional PoPs, there are frontside and backside RDLs in a single-chip InFO PoP for signal redistributions, as shown in Figure 1(a). To enhance compatibility, the extra backside RDL can rearrange signals from top-package

This work was partially supported by AnaGlobe, Genesys Logic, IBM, MediaTek, TSMC, Academia Sinica, MOST of Taiwan under Grant MOST 103-2221-E-002-259-MY3, MOST 104-2221-E-002-132-MY3, MOST 105-2221-E-002-190-MY3, and MOST 106-2911-I-002-511.

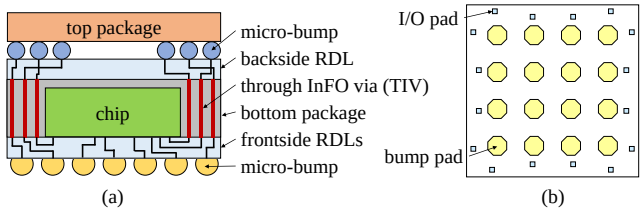


Fig. 1. (a) A cross-sectional view of a single-chip InFO PoP. (b) A typical RDL structure of traditional flip-chip designs with peripheral I/Os.

micro-bumps to TIVs and transform an areal micro-bump array to peripheral TIVs [16]. In modern IC designs, I/O pads are usually placed along the boundary of a chip, named *peripheral I/Os*. The frontside RDLs are used to redistribute I/O pads to bump pads or connect TIVs to bump pads or I/O pads. Thus, it is desirable to develop an RDL router to complete the routing on both the backside and frontside RDLs and utilize the freedom of assigning TIVs to connect the two RDLs. The frontside RDLs in a single-chip InFO PoP are divided into two regions: (1) the fan-in region, which is the region directly underneath a chip, and (2) the fan-out region, which is the region not belonging to the fan-in region [9].

A. Previous Works

The traditional flip-chip designs usually contain one chip with no fan-out region in the corresponding RDL structure, as shown in Figure 1(b). Depending on the interactions among IC, packaging, and PCB designers, the traditional RDL routing problem can be categorized into three major types: (1) the free-assignment routing problem, (2) the pre-assignment routing problem, (3) the unified-assignment routing problem [8]. First, for the free-assignment routing problem, the connections between I/O pads and bump pads are not predefined before routing. Second, for the pre-assignment routing problem, the connections between I/O pads and bump pads are predefined by designers. Third, for the unified-assignment routing problem, some assignments between I/O pads and bump pads are predefined and the rest of assignments are not. To deal with these three types of RDL routing problems for the conventional flip-chip designs, various works have been published. The works [5, 12] addressed the free-assignment routing problem. Fang *et al.* [5] adopted a network-flow-based algorithm to obtain a global routing solution. Liu *et al.* [12] also developed a network-flow-based approach based on Delaunay triangulation and Voronoi diagram. The works [4, 10] focused on the pre-assignment routing problem. Fang *et al.* [4] presented an integer-linear-programming-based method with reduction techniques to accomplish the global routing in a reasonable runtime. Lin *et al.* [10] proposed a ring-by-ring routing algorithm from

the innermost bump-pad ring to the outer bump-pad rings. In each iteration, this router exchanges a net sequence based on an efficient dynamic programming algorithm. Fang *et al.* [6] addressed the unified-assignment routing problem. In this work, the area I/O structure is considered; that is, an I/O pad can be placed at any location. They presented an algorithm including three major stages: (1) apply a network-flow-based method to derive the congestion information of free-assignment connections, (2) iteratively route the pre-assignment nets considering the congestion information, and (3) route the free-assignment nets on a modified flow network considering the routed pre-assignment nets.

In addition to the traditional RDL routing problems for the flip-chip designs, Lin *et al.* [9] tackled the unified-assignment, multi-layer multi-chip RDL routing problem for the InFO package. In this work, multiple chips share the same RDLs, which are used to redistribute I/O pads to bump pads or connect I/O pads among different chips. They proposed a concentric-circle model to model all the connections among one chip and all other chips. Based on the model, the connections between chips were assigned to appropriate layers and a modified flow network was applied to obtain a good global routing prototype.

B. Differences from the Previous Works

The InFO PoP designs have frontside and backside RDLs for signal redistributions. The frontside and backside RDL structures are shown in Figures 3(a) and (b), respectively. Different from the RDL structures of previous works, there are TIVs (TIV pads) on both the frontside and backside RDL structures to provide signal paths between the two RDLs. In addition, the types of connections are more complicated than the previous works. There are three types of connections: (1) connection from a top-package micro-bump to a TIV and then a bottom-package micro-bump, (2) connection from a top-package micro-bump to a TIV and then an I/O of a chip, and (3) connection from an I/O of a chip to a bottom-package micro-bump. In our addressed problem, the second type of connections are predefined between top-package micro-bumps and I/Os of a chip (the connections of TIVs are not predefined), because the functions of I/Os of a chip and top-package micro-bumps are typically predefined by designers.[1] To tackle this new problem, we could have utilized and extended the work [10] for the backside RDL routing and the work [9] for the frontside RDL routing. However, this extension may suffer from a severe routing solution degradation, as shown in Figures 2(a) and (b). The reason there exists the net on an additional layer is that their layer-assignment method of iteratively finding longest common subsequences (LCS's) cannot guarantee to obtain the optimal layer-assignment solution. The reason there exists the detour is that they do not consider the congestion of free-assignment nets nearby TIVs. As a result, it is desirable to develop an RDL router considering the interactions between the top package, the chip in the bottom package, and the TIVs. Therefore, we propose a new algorithm considering the aforementioned interactions to achieve better routing solution, as shown in Figures 2(c) and (d).

C. Our Contributions

We summarize our main contributions as follows:

[1]For practical applications, we can treat the top package as a black box because it is usually provided by another party, and typically its connections between I/Os of a chip and micro-bumps cannot be changed. Thus, the functions of top-package micro-bumps are predefined.

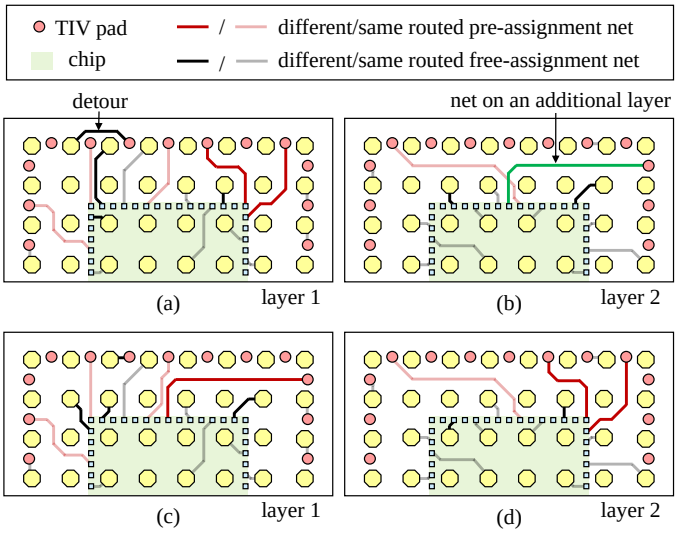


Fig. 2. The comparison between the previous works [9, 10] with extensions and our algorithm. (a)(b) A three-layer frontside RDL routing result derived from the previous works [9, 10] with extensions. The reason there exists the net on an additional layer is that their layer-assignment method of iteratively finding longest common subsequences (LCS's) cannot guarantee to obtain the optimal layer-assignment solution. The reason there exists the detour is that they do not consider the congestion of free-assignment nets nearby TIVs. (c)(d) A much better two-layer frontside RDL routing result derived from our algorithm.

- This paper is the first work in the literature to formulate the unified-assignment multi-layer multi-package RDL routing problem for the wafer-level integrated fan-out (InFO) package-on-package (PoP) and present the first algorithm to resolve this problem.
- We propose a ring-by-ring routing algorithm based on extracting increasing subsequences (IS). We transform a routing sequence into two directed acyclic graphs (DAGs), IS-DAG and Constraint-DAG. To minimize the layer number of the frontside RDLs, we minimize the number of vertices on the longest path on the Constraint-DAG. Moreover, we perform backtracking on the IS-DAG to efficiently assign the connections to appropriate layers to avoid long detours.
- We present a network-flow-based approach to perform TIV/layer assignment for the backside RDL routing. By considering the congestion information on frontside RDLs, better TIV/layer-assignment results can facilitate the following frontside RDL routing to effectively minimize the number of detours of free-assignment nets.
- We propose a free-assignment nets pad/layer reassignment method for the frontside RDL routing. We observe that pre-assignment nets between I/O pads and TIVs divide the fan-out region into several fan-out sub-regions. For each fan-out sub-region on each layer, we apply a bipartite matching algorithm to further reduce the number of detours of pre-assignment nets caused by free-assignment nets.
- Experimental results show that our algorithm is effective and efficient. In particular, our router can achieve 100% routablility for all test cases, while the previous works with extensions fail to complete the routing even with more frontside RDLs than ours.

The rest of this paper is organized as follows. Section II formulates the addressed problem and provides the routing design

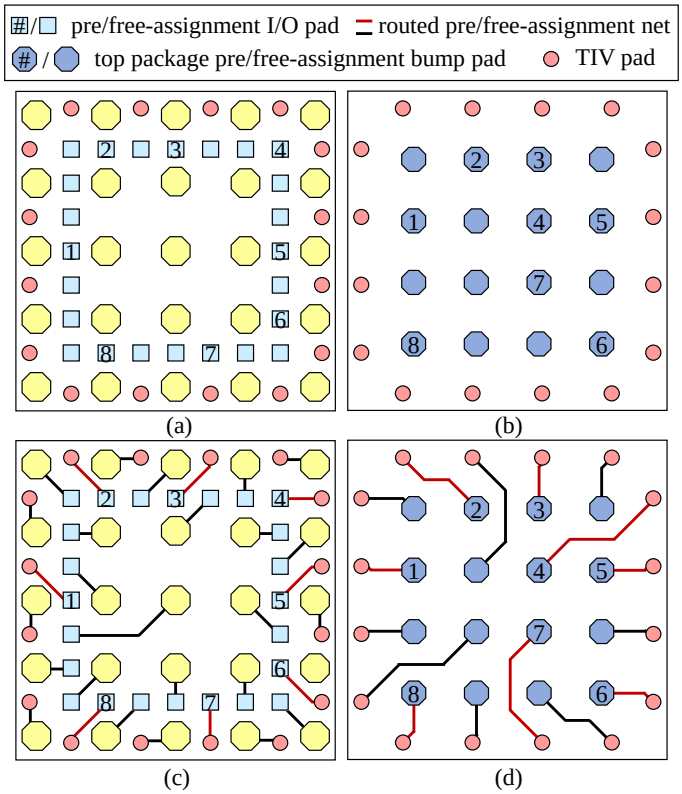


Fig. 3. (a) The structure of the considered frontside RDL routing plane. (b) The structure of the considered backside RDL routing plane. (c) A single-layer frontside RDL routing result corresponding to (a). (d) A single-layer backside RDL routing result corresponding to (b).

rules. Section III details our proposed algorithm. Section IV shows the experimental results. Section V concludes this paper.

II. Preliminaries

In this section, we formally define the new RDL routing problem and detail the routing design rules corresponding to our problem.

A. Problem Formulation

In this paper, we deal with the unified-assignment multi-layer multi-package RDL routing problem, which is different from the previous works, as mentioned in Section I-B. We first give the following terminologies and notations used throughout this paper:

- *Top-package free-assignment bump pad:* a top-package bump pad that has no predefined assignment.
- *Top-package pre-assignment bump pad:* a top-package bump pad that has a predefined assignment to a certain I/O pad.
- *Free-assignment I/O pad:* an I/O pad that has no predefined assignment.
- *Pre-assignment I/O pad:* an I/O pad that has a predefined assignment to a certain top-package bump pad.
- *Bottom-package fan-in bump pad:* a bottom-package bump pad located in the fan-in region.
- *Bottom-package fan-out bump pad:* a bottom-package bump pad located in the fan-out region.
- *TIV pad:* a TIV corresponding to two TIV pads (one on the frontside RDLs and another on the backside RDL)
- $B_t = \{b_1^t, b_2^t, ..., b_j^t\}$ is the set of all top-package bump pads, where j is the number of the top-package bump pads.
- $B_b = \{b_1^b, b_2^b, ..., b_k^b\}$ is the set of all bottom-package bump pads, where k is the number of the bottom-package bump pads.
- $Q = \{q_1, q_2, ..., q_u\}$ is the set of all I/O pads, where u is the number of I/O pads.
- $N = \{n_1, n_2, ..., n_m\}$ is the set of pre-assignment nets, where m is the number of pre-assignment nets and n_i defines the connection between a top-package bump pad and an I/O pad.

The structure of the considered frontside and backside RDL routing planes are shown in Figures 3(a) and (b), respectively. Different from the previous works, there are TIVs to provide the connections between the frontside and backside RDLs. In addition, there are some predefined I/O pads and top-package bump pads that need to be connected in the final routing solution. Figures 3(c) and (d) show the single-layer routing results satisfying all the design rules corresponding to Figures 3(a) and (b). According to the work [14], we confine the layer number of the backside RDL to one, and intend to minimize the layer number of the frontside RDLs. Here, we formally define the problem as follows:

- **The Unified-Assignment Multi-Layer Multi-Package RDL Routing Problem:** Given a frontside RDL layout, a backside RDL layout, pre-assignment netlists N, and design rules, connect all pre-assignment nets $n_i \in N$ and connect all the other I/O pads and top-package bump pads to some bottom-package bump pads $b^b \in B_b$ so that there is no net crossing and the total wirelength and the number of frontside RDLs are minimized.

B. Routing Design Rules

In this paper, the most popular peripheral I/O structure is adopted. All the I/O pads of a chip in the bottom package are placed along the chip boundary. To ease the design complexities or fabrication difficulties, our router should handle several routing design rules. The three major design rules in our problem are detailed as follows:

- *Routing-region constraint:* Detours/multiple crossings of a net on the chip boundary are not allowed because the distribution of I/O pads on the chip boundary is very dense [9].
- *Non-crossing constraint:* Net crossing is not allowed on the same layer. Moreover, for better signal integrity, each route should be completed on a single layer, and vias (not the TIVs) are often not allowed (bump pads and I/O pads are punched through all RDLs) [3].
- *Routing-angle constraint:* Our router can perform routing angles of 90-degree or 135-degree. However, a routing angle of 45-degree is not allowed.

III. Algorithms

In this section, we present our routing algorithm. First, we provide an overview of our proposed algorithm. Then, we detail the methods used in each stage of the algorithm in the following subsection.

A. Algorithm Overview

Figure 4 summarizes our routing algorithm, which consists of three major stages: (1) congestion estimation on frontside RDLs, (2) backside RDL global routing, and (3) frontside RDL global routing.

In the first stage, to estimate the routing congestion on frontside RDLs, we first perform escape routing to derive a

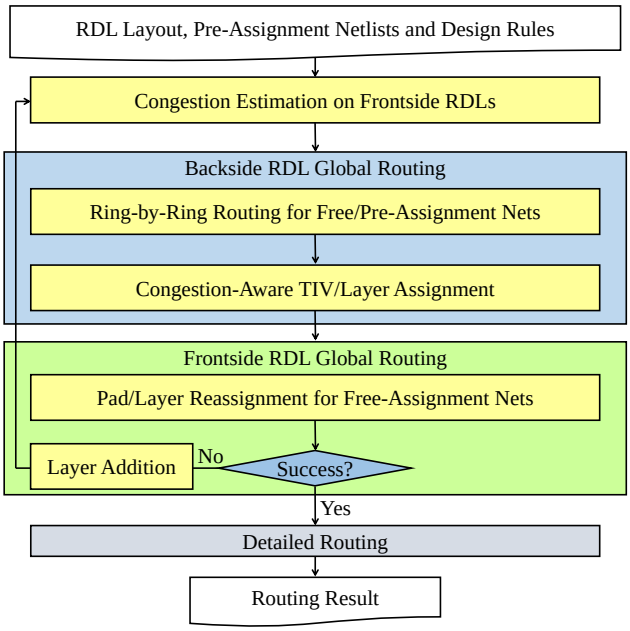


Fig. 4. Routing flow of our proposed algorithm.

routing prototype for bottom-package fan-in bump pads, and then construct a flow network to assign free-assignment I/O pads to bottom-package fan-out bump pads and escaped routes from escape routing. We collect the number of adjacently assigned bottom-package bump pads on each layer for each TIV as the corresponding congestion information. In the second stage, we first apply an iterative outward ring-by-ring routing method to route all top-package bump pads to the outermost top-package bump-pad ring. With the corresponding congestion information of each TIV, we then perform a flow-network based method to assign top-package bump-pad routes to some proper TIV pads to avoid potential routing congestion. In the third stage, we reassign some free-assignment nets by bipartite matching to further improve our routing solution.

In the stage of detailed routing, we apply the methods in [3] and [10] to derive the final detailed-routing solution.

B. Congestion Estimation on Frontside RDLs

To estimate the congestion on frontside RDLs, we first perform escape routing to obtain a routing prototype (escaped routes) in the fan-in region, and then assign all free-assignment I/O pads to some bottom-package fan-in bump pads (the escaped routes) or some bottom-package fan-out bump pads. 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 [15]. We implement the escape routing algorithm proposed in [15] to derive the escaped routes. Next, to solve the free-assignment problem from the I/O pads to the bottom-package bump pads, which also has been well-studied recently, we apply the methods proposed in [9], where the minimum-cost maximum-flow (MCMF) algorithm [1] is adopted to obtain a desired assignment topology. Finally, for each TIV, the number of adjacently assigned bottom-package bump pads on each layer are collected as the congestion information on each layer, which can facilitate the backside RDL routing to avoid undesired routing results in the following stages.

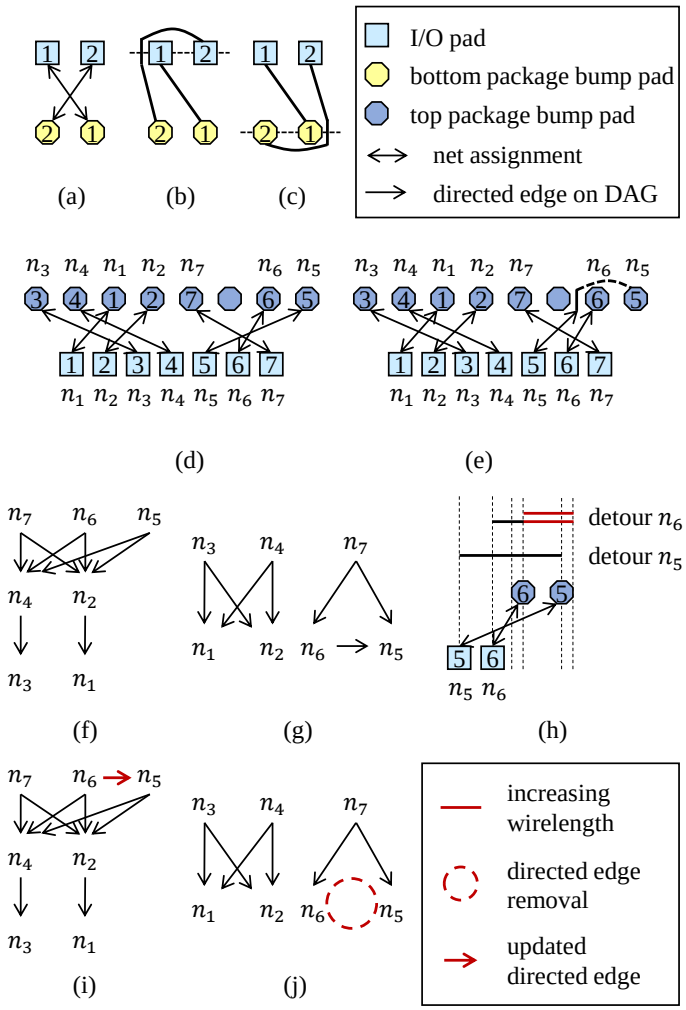


Fig. 5. (a) Given non-consistent I/O pad and bump-pad sequences, a crossing will be induced by directly connecting the I/O pads and bump pads without detours. (b) An exchanged net sequence along the I/O pad ring. (c) An exchanged net sequence along the bump-pad ring. (d) The transformed sequences structure. The I/O pad sequence is $S_{io} = (n_1, n_2, n_3, n_4, n_5, n_6, n_7)$, and the top-package bump-pad sequence is $S_{tb} = (n_3, n_4, n_1, n_2, n_7, n_6, n_5)$, where n_i represents the net ID. (e) The sequence structure with the detour. After removing the directed edge from n_6 to n_5 on the Constraint-DAG (g), we need to detour on the top-package bump-pad ring. (f)(g) The corresponding IS-DAG and Constraint-DAG, respectively, constructed from the top-package bump-pad sequence $S_{tb} = (n_3, n_4, n_1, n_2, n_7, n_6, n_5)$. (h) To remove the directed edge from n_6 to n_5 on the Constraint-DAG (g), we can either detour n_6 to the right side of n_5 or detour n_5 to the left side of n_6. In this example, n_5 is detoured to the left side of n_6 because it has a smaller cost. (i)(j) The updated IS-DAG and Constraint-DAG, respectively. After removing the directed edge from n_6 to n_5 on the Constraint-DAG (g), the IS-DAG (f) needs to be updated.

C. Backside RDL Global Routing

In this section, we detail our backside RDL global routing method, considering the routing among top-package bump pads and TIV pads. All the top-package bump pads need to be connected to some TIV pads in order to reach the bottom-package bump pads and I/O pads on the frontside RDLs.

1) Ring-by-Ring Routing for Free/Pre-Assignment Nets: As mentioned in Section I-B, our ring-by-ring routing algorithm is inspired by [10], but with key differences. Here, we first make a brief review on the *sequence exchange* concept proposed in [10].

As shown in Figure 5(a), given non-consistent I/O pad and bump-pad sequences, a crossing will be induced by directly connecting the I/O pads and bump pads without detours. To finish the routing, the net sequence must be exchanged either along the I/O pad ring (see Figure 5(b)) or along the bump-pad ring (Figure 5(c)). In this paper, we only exchange the net sequence along the top-package bump-pad ring. Because the distributions of I/O pads and TIV pads are very dense, we cannot perform detours (exchange the net sequence) along the I/O pad ring or TIV pad ring. Based on the sequence exchange concept and our extracting increasing subsequences (IS) algorithm, we perform the ring-by-ring routing from the innermost top-package bump-pad ring to the outer top-package bump-pad rings.

For each iteration in the ring-by-ring routing, an outer top-package bump-pad ring is referred to as the current ring, and the inner top-package bump-pad ring in the previous iteration is referred to as the preceding ring. To perform ring-by-ring routing, we apply our extracting increasing subsequences (IS) algorithm on the current ring, and the maximum planar subset of chords (MPSC) algorithm [13] on the current and preceding rings. The objective of our extracting IS algorithm at this stage is to decide whether a net on the current ring needs to detour or not. Besides, our extracting IS algorithm can assign the pre-assignment nets to appropriate layers of frontside RDLs for the stage of congestion-aware TIV/layer assignment, to be discussed in the next subsection.

To apply our extracting IS algorithm, we first transform the ring structure into a sequence structure by finding a cut line without crossing any net between rings. If there does not exist such a cut line, we find a cut line crossing with minimal number of nets, and remove these crossed nets to become an extra increasing subsequence. For the transformed I/O pad and top-package bump-pad sequences, we first label the I/O pad sequence with increasing ordered net IDs, and then derive the corresponding I/O pad and top-package bump-pad sequences with net IDs. For example, as shown in Figure 5(d), the I/O pad sequence is $S_{io} = (n_1, n_2, n_3, n_4, n_5, n_6, n_7)$, and the top-package bump-pad sequence is $S_{tb} = (n_3, n_4, n_1, n_2, n_7, n_6, n_5)$, where n_i represents the net ID. Note that we skip the top-package free-assignment bump pads here. As shown in Figure 5(d), the top-package bump pad without any labeled net ID is the free-assignment one, which we do not consider in the top-package bump-pad sequence. The reason is that the top-package free-assignment bump pads have no predefined assignments, and thus they always can be directly routed without any detour between the current and preceding rings by the MPSC algorithm.

Based on a top-package bump-pad sequence, we can construct two directed acyclic graphs (DAGs), namely, *IS-DAG* and *Constraint-DAG*. Every net ID in the top-package bump-pad sequence corresponds to a vertex on the IS-DAG and Constraint-DAG, respectively. To construct the directed edges, we traverse from the beginning to the end of a top-package bump-pad sequence. For each traversed net ID, if any previous net ID is smaller than the currently traversed net ID, a directed edge on the IS-DAG is constructed from the vertex for the currently traversed net ID to the vertex for the previous net ID; otherwise, a directed edge on the Constraint-DAG is constructed from the vertex for the previous net ID to the vertex for the currently traversed net ID. For example, the corresponding IS-DAG and Constraint-DAG of $S_{tb} = (n_3, n_4, n_1, n_2, n_7, n_6, n_5)$ are shown in Figures 5(f) and (g), respectively. Note that for better illustration, some directed edges with the transitive closure property are not shown in Figure 5(f).

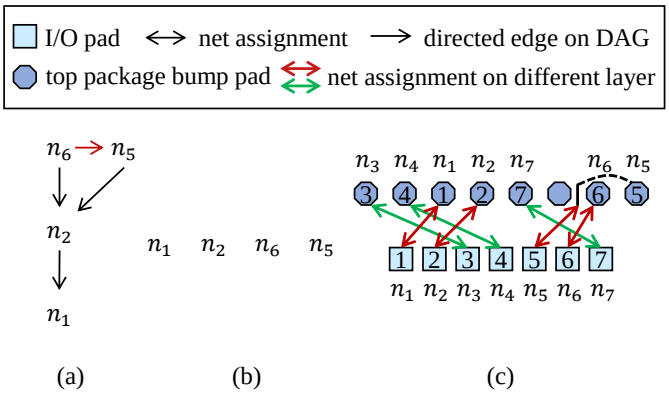


Fig. 6. An example of our extracting IS algorithm allocating the pre-assignment nets to appropriate layers. (a) and (b) The updated IS-DAG and Constrain-DAG, respectively, after the first iteration of backtracking. (c) The sequence structure with the net assignment on different layers of the frontside RDLs.

We observe that the number of vertices on the longest path on the Constraint-DAG implies the number of derived increasing subsequences, also, the number of layers used in the layer assignment of pre-assignment nets. Therefore, it is desirable to minimize the number of vertices (by removing some edges) on the longest path on the Constraint-DAG to further minimize the layer number. However, removing an edge on the Constraint-DAG could introduce a sequence exchange (detour) on the top-package bump-pad ring, so we need to find a better trade-off between layer-number and wirelength minimization. Practically, there may be several longest paths (with the same number of vertices) on the Constraint-DAG, and thus, for each longest path we will only remove an edge (detour) once. The longest path in a DAG problem can be solved in linear time with the topological sorting technique [2]. After finding the longest path, we calculate the costs of each directed edge on the longest path and remove the directed edge with minimal cost. There are two costs for each directed edge because the detour could be bidirectionally, and we define the cost as the increasing wirelength to make a detour. As shown in Figures 5(g) and (h), to remove the directed edge from n_6 to n_5, we can either detour n_6 to the right side of n_5 or detour n_5 to the left side of n_6. In this example, n_5 is detoured to the left side of n_6 because it has a smaller cost. Figure 5(j) shows an example of removing an directed edge from n_6 to n_5 on the longest path $P = (n_7, n_6, n_5)$, and Figure 5(e) shows the corresponding detour on the top-package bump-pad ring. After removing a directed edge on the Constraint-DAG, the IS-DAG needs to be updated to reflect the detour on the top-package bump-pad ring, as shown in Figure 5(i).

Once the last iteration of the ring-by-ring routing is completed, all of the top-package bump pads have been routed to the outermost top-package bump-pad ring. The IS-DAG and the Constraint-DAG in the final iteration will be preserved to perform the extracting IS algorithm to allocate the pre-assignment nets to appropriate layers of frontside RDLs at the next stage.

2) Congestion-Aware TIV/Layer Assignment: After the ring-by-ring routing for free/pre-assignment nets, all of the top-package bump pads are already routed to the outermost top-package bump-pad ring, as shown in Figure 7(a). We refer to these routed top-package bump pads as *top-package bump-pad routes*. In addition, we refer to the outermost top-package bump

pads of bump-pad arrays as *boundary top-package bump pads*. At this stage, to complete the backside RDL global routing, we need to connect each top-package bump-pad route and each boundary top-package bump pad to an appropriate TIV pad, and assign the same-location TIV pad on the frontside RDLs to a proper layer. Based on the extracting IS algorithm and the IS-DAG and Constraint-DAG preserved at the previous stage, we first sequentially assign the pre-assignment nets (the top-package pre-assignment bump-pad routes and the boundary top-package pre-assignment bump pads) to some TIV pads to maintain the sequence order, and decide the routing layer of the same-location TIV pads on the frontside RDLs. Then, we construct a flow network and apply the minimum-cost maximum-flow (MCMF) algorithm [1] to simultaneously assign the free-assignment nets (the top-package free-assignment bump-pad routes and the boundary top-package free-assignment bump pads) to TIV pads and the routing layer of the same-location TIV pads on the frontside RDLs. Note that the TIVs provide the connections between the frontside and backside RDLs. To reduce the routing congestion on the frontside RDLs, we integrate the congestion information into our network-flow model to consider the frontside RDL and backside RDL routing simultaneously.

The extracting IS algorithm is applied to assign the pre-assignment nets on proper layers of frontside RDLs. We refer to the vertices on the Constraint-DAG that have no incoming edges as *free vertices*; otherwise, *constrained vertices*. As shown in Figure 5(j), n_3, n_4, and n_7 are free vertices, and n_1, n_2, n_6, and n_5 are constrained vertices. To extract increasing subsequences, we perform iterative backtracking on the IS-DAG. In each iteration, the backtracking starts from a vertex on the IS-DAG that has no outgoing edges. In addition, during backtracking, we also ensure that each backtracked vertex has a same-net-ID free vertex on the Constraint-DAG and has minimal difference of net ID from the previously backtracked vertex. As shown in Figure 5(i), we start backtracking from the vertex n_3 because the same-net-ID vertex of n_1 on the Constraint-DAG (see Figure 5(j)) is a constrained vertex. The increasing subsequence derived from backtracking on the IS-DAG in Figure 5(i) is $S_{is1} = (n_3, n_4, n_7)$. After each iteration of backtracking is completed, the IS-DAG and Constrain-DAG will be updated by removing the backtracked vertices. Figures 6(a) and (b) show the updated IS-DAG and Constrain-DAG, respectively, after the first iteration of backtracking.

Once the last iteration of backtracking is completed, we have extracted all increasing subsequences on the IS-DAG. For example, the increasing subsequence derived from the backtracking on the updated IS-DAG in Figure 6(a) is $S_{is2} = (n_1, n_2, n_5, n_6)$. As a result, all of the increasing subsequences extracted from the original IS-DAG in Figure 5(i) will be $A_{is\_total} = \{(n_3, n_4, n_7), (n_1, n_2, n_5, n_6)\}$. We assign different layers of the frontside RDLs for each extracted increasing subsequence to avoid detours or net crossings on the frontside RDLs, as shown in Figure 6(c). Finally, we sequentially connect each pre-assignment net to its nearest TIV pad to maintain the sequence order, as shown in Figure 7(b). For each connected TIV pad, the same-location TIV pad on the frontside RDLs is assigned to the routing layer according to the assigned layer of the pre-assignment net in increasing subsequences.

On the other hand, we construct a flow network to deal with the free-assignment nets. Because the free-assignment nets will eventually be connected to some bottom-package bump pads, we integrate the congestion information into the flow network to consider the routing-resource allocation of the bottom-package

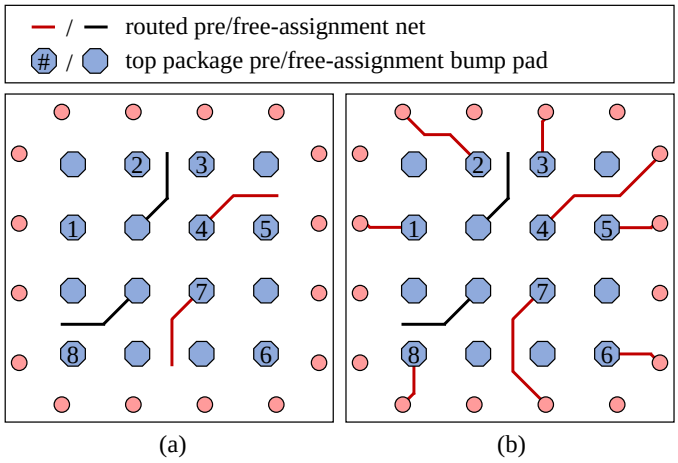


Fig. 7. (a) A ring-by-ring routing example from Figure 3(b). All of the top-package bump pads are routed to the outermost top-package bump-pad ring. (b) A TIV assignment example. Each of the pre-assignment net is connected to a TIV pad.

bump pads. Here, we introduce the construction of the flow network. Each free-assignment net corresponds to a vertex in the flow network. For each non-assigned TIV pad, we create L duplicated vertices to represent the same-location TIV pad on different frontside RDLs, where L is the layer number of the frontside RDLs currently. Besides, we create a super vertex for each non-assigned TIV pad to ensure that each non-assigned TIV pad will be only assigned on one layer. Finally, two super vertices are created as the source vertex and sink vertex in the flow network. There are four types of directed edges:

- Edge from the source vertex to the vertex for a free-assignment net.
- Edges from the vertex for a free-assignment net to the L duplicated vertices for a neighboring, non-assigned TIV pad.
- Edges from the L duplicated vertices for a non-assigned TIV pad to the super vertex for the same TIV pad.
- Edge from the super vertex for a non-assigned TIV pad to the sink vertex.

Every edge has a unit capacity. The cost of the second type of edges includes the cost of wirelength C_{wl}, the cost of congestion C_{cong}, and the cost of crossing C_{cros}.

$$cost = \alpha C_{wl} + \beta C_{cong} + C_{cros}, \tag{1}$$

where α and β are user-defined parameters, C_{wl} is the Euclidian distance between the TIV pad and the boundary top-package free-assignment bump pad or the bump-pad array boundary location of the top-package free-assignment bump-pad route, and C_{cong} is the congestion information of the TIV pad, which is collected in Section III-B. Note that each L duplicated vertex for a TIV pad has a different estimated congestion according to its represented layer on the frontside RDLs. C_{cros} is the cost to avoid crossings with previous connected pre-assignment nets to some TIV pads. If there exist such crossings, we give C_{cros} a large value; otherwise, C_{cros} is zero. The cost of all the other edges is zero.

After applying the MCMF algorithm on the flow network, each of the top-package bump-pad route and boundary top-package bump pad is assigned to a proper TIV pad, as shown in Figure 3(d), and the same-location TIV pad on the frontside RDLs is assigned on an appropriate layer.

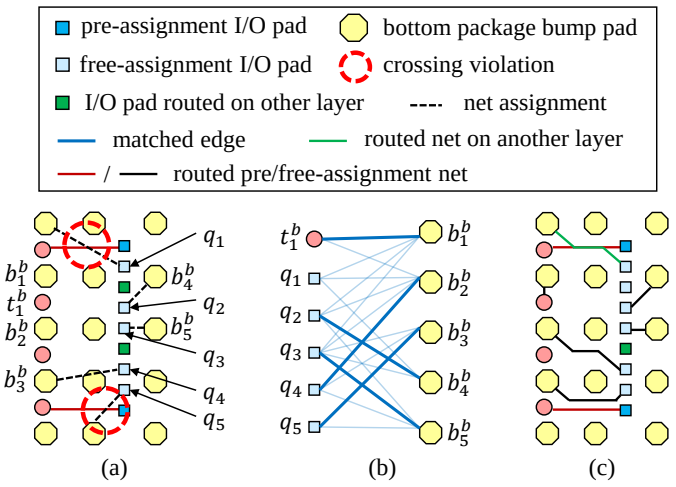


Fig. 8. An example for our pad/layer reassignment for free-assignment nets. (a) The fan-out sub-region is within two pre-assignment nets, and there are two crossing violations that need to be resolved. (b) The constructed bipartite graph corresponds to (a). (c) A routing solution based on the result of the bipartite matching algorithm in (b).

D. Frontside RDL Global Routing

In this section, we detail our frontside RDL global routing method, considering the routing among I/O pads, bottom-package bump pads, and TIV pads. All the free-assignment I/O pads and the TIV pads connected by the top-package free-assignment bump pads need to be connected to some bottom-package bump pads. In addition, each of the pre-assignment I/O pad should be connected to the TIV pad, where the same-location TIV pad on the backside RDL is connected by the corresponding top-package pre-assignment bump pad. In other words, each pair of the top-package pre-assignment bump pad and the pre-assignment I/O pad should be connected to exactly the same TIV.

1) Pad/Layer Reassignment for Free-Assignment Nets: After the congestion estimation on frontside RDLs, a desired assignment topology of all the free-assignment I/O pads is derived. In addition, at this stage, the backside RDL global routing is completed, and hence we have the assignment and layer information of all the pre-assignment nets (between the pre-assignment I/O pads and the TIV pads). Also, the assignment of the TIV pads (TIVs connected by top-package free-assignment bump pads) to some bottom-package bump pads are yet to be determined (the routing layers have been decided in the previous stage, in the Section III-C2). We refer to these TIV pads as *free-assignment TIV pads*. Besides, we refer to the outermost bottom-package bump pads of bump-pad arrays inside the chip to *boundary bottom-package bump pads*. As a result, we propose a pad/layer reassignment algorithm for free-assignment nets by bipartite matching to fix the net crossing as well as assign all the free-assignment nets (the free-assignment I/O pads and the free-assignment TIV pads) to some bottom-package bump pads.

We observe that the pre-assignment nets between I/O pads and TIV pads divide the fan-out region into several fan-out sub-regions. Therefore, for each fan-out sub-region on each layer, a *free-assignment sub-problem* is formed, and we solve these free-assignment sub-problems based on a region-by-region, layer-by-layer scheme. As shown in Figure 8(a), the fan-out sub-region is within two pre-assignment nets, and there are two crossing violations that need to be resolved. In addition, the free-assignment TIV pad t_1^b needs to be connected to a bottom-package bump pad. To resolve the free-assignment sub-problem, we construct a bipartite graph $G_{bm} = (V_{bm}, E_{bm})$, where V_{bm} is a set of vertices and E_{bm} is a set of edges. In the fan-out sub-region, every bottom-package fan-out bump pad assigned on the current layer and every unassigned one correspond to a vertex. In the fan-in region, every neighboring escaped route and neighboring boundary bottom-package bump pad correspond to a vertex. Besides, every free-assignment TIV pad and I/O pad on the current layer correspond to a vertex. There are three types of edges:

- Edge between the vertex for a free-assignment I/O pad and that for a bottom-package fan-out bump pad or a neighboring boundary bottom-package bump pad.
- Edge between the vertex for a free-assignment I/O pad and that for a neighboring escaped route.
- Edge between the vertex for a free-assignment TIV pad and that for a bottom-package fan-out bump pad.

As shown in Figure 8(b), the constructed bipartite graph corresponds to the free-assignment sub-problem in Figure 8(a), where the escaped routes are not shown for better illustration. The weights of the first and third types of edges are the Euclidian distance between the corresponding pads. The weight of the second type of edge is the wirelength of the escaped route plus the Euclidian distance between the I/O pad and the bottom-package fan-in bump-pad array boundary location of the escaped route. Then, we perform the minimum-weight maximum-cardinality bipartite-matching algorithm [7]. Figure 8(c) shows the final routing solution based on the result of the bipartite-matching algorithm in Figure 8(b). Note that if there exist unmatched free-assignment I/O pads or TIV pads, we have to reassign them to another layer or detour some pre-assignment nets.

IV. Experimental Results

TABLE I
Benchmark circuit statistics.

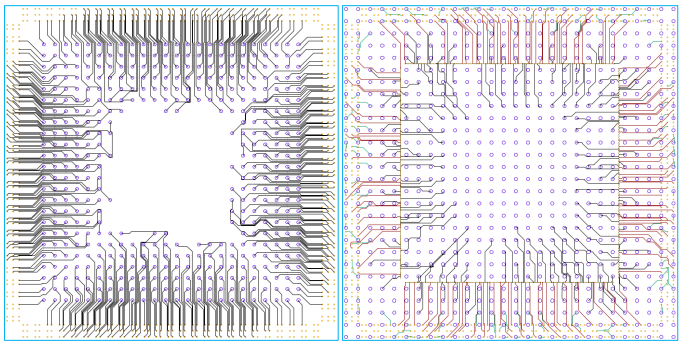
Circuits	#TIVs	$\|B_t\|$	$\|Q\|$	$\|B_b\|$	$\|N\|$
infoPoP1	552	393	484	676	209
infoPoP2	600	451	636	784	216
infoPoP3	696	568	849	1024	268
infoPoP4	840	692	1089	1444	320
infoPoP5	1032	868	1252	2116	396

We implemented our algorithm in the C++ programming language. All of the experiments were performed on an Intel Xeon 2.93GHz Linux machine with 48GB memory. Table I lists the benchmark circuits. "#TIVs", "$|B_t|$", "$|Q|$", "$|B_b|$", and "$|N|$" denote the numbers of TIVs, top-package bump pads, I/O pads, bottom-package bump pads, and pre-assignment nets, respectively.

We compared our algorithm with two RDL routing flows, namely Ft-first and Bk-first. Both Ft-first and Bk-first utilize the work [10] for the backside RDL routing and the work [9] for the frontside RDL routing, but Ft-first routes the frontside RDLs first and Bk-first routes the backside RDL first. To extend the work [9] to handle the frontside RDL routing, we treat the TIV pads on the frontside RDLs as four sets of I/O pads that belong to four virtual chips along the package boundary. To apply the work [10] to deal with the backside RDL routing, we treat the

TABLE II
COMPARISON OF THE OVERALL RDL ROUTING RESULTS BETWEEN FT-FIRST, BK-FIRST, AND OURS ("N/A": INCOMPLETE ROUTING RESULTS).

Circuits	#Frontside RDLs			Routability (%)			Total Wirelength (μm)			Runtime (sec.)		
	Ft-first	Bk-first	Ours	Ft-first	Bk-first	Ours	Ft-first	Bk-first	Ours	Ft-first	Bk-first	Ours
infoPoP1	3	3	2	81.27	92.52	100.00	N/A	N/A	463440	27	29	22
infoPoP2	3	3	2	74.90	93.69	100.00	N/A	N/A	740623	38	48	36
infoPoP3	3	3	2	83.04	95.04	100.00	N/A	N/A	889535	96	122	78
infoPoP4	3	3	2	78.06	95.28	100.00	N/A	N/A	1020876	315	375	214
infoPoP5	3	3	2	82.57	94.55	100.00	N/A	N/A	1240019	1053	1193	637
Comparison	-	-	-	0.80	0.94	1.00	-	-	-	1.55	1.79	1.00



(a) (b)

Fig. 9. The RDL routing solution of infoPoP2. (a) The backside RDL routing solution. (b) First layer of the frontside RDL routing solution. The red lines are the pre-assignment nets from I/O pads to TIV pads, the black lines are the free-assignment ones from I/O pads to bottom-package bump pads, and the green lines are the free-assignment ones from TIV pads to bottom-package bump pads.

TIV pads on the backside RDL as I/O pads. If Ft-first or Bk-first cannot complete the routing under the current number of layers, we add one more frontside RDL and start over. According to the work [14], the maximum number of frontside RDLs are set to three.

Table II shows the experimental results and reports the number of frontside RDLs, the routability, the total wirelength, and the runtime. As shown from the table, compared with Ft-first and Bk-first, our algorithm can achieve 100% routability for all the five benchmarks by using only two frontside RDLs, while Ft-first and Bk-first fail to obtain a routing solution for any of the benchmarks by using even three frontside RDLs. The results show that our algorithm is effective and efficient for the InFO PoP routing.

V. CONCLUSIONS

In this paper, we have presented an RDL routing algorithm for the InFO PoPs. In addition, we are the first in the literature to formulate the unified-assignment multi-layer multi-package RDL routing problem for the InFO PoPs. Based on our extracting IS algorithm, we have efficiently minimized the layer number of the frontside RDLs and effectively assigned the connections to appropriate layers. To utilize the freedom of assigning TIVs, we have proposed a network-flow-based TIV/layer-assignment approach for the free-assignment nets considering the interactions among the two RDLs and packages to effectively minimize the number of detours of free-assignment nets. For the frontside RDL routing, we have presented a free-assignment nets pad/layer reassignment method to reduce the number of detours of pre-assignment nets caused by free-assignment nets. The experimental results have demonstrated the efficiency and effectiveness of our proposed algorithm.

REFERENCES

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

[2] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT press, 2009.

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

[8] 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.

[9] 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 ICCAD*, pages 1–8, November 2016.

[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] K. J. Supowit. Finding a maximum planar subset of a set of nets in a channel. *IEEE TCAD*, 6(1):93–94, January 1987.

[14] C.-F. Tseng, C.-S. Liu, C.-H. Wu, and D. Yu. InFO (wafer level integrated fan-out) technology. In *Proc. of ECTC*, pages 1–6, May 2016.

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

[16] 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.