Recent Research Development in Flip-Chip Routing

Hsu-Chieh Lee[†], Yao-Wen Chang[†‡], and Po-Wei Lee[†]
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan[†]
Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan [‡]
pg30123@eda.ee.ntu.edu.tw; ywchang@cc.ee.ntu.edu.tw; webber@eda.ee.ntu.edu.tw

***Abstract*—The flip-chip package is introduced for modern IC designs with higher integration density, larger I/O counts, faster speed, better signal integrity, etc. To ease design changes, an extra metal layer is introduced to redistribute nets between wire-bonding (I/O) pads in a die and bump pads in a package carrier. Flip-chip routing is performed by redistributing and interconnecting nets between the I/O and bump pads. As the design complexity grows, routing has played a pivotal role in flip-chip design. In this paper, we first introduce popular flip-chip structures, their routing-region modeling, and induced routing problems, survey key published techniques for flip-chip routing with respect to specific structures and pad assignment methods, and provide some future research directions for the modern flip-chip routing problem.**

I. Introduction

For modern IC designs, the increasing design complexity and decreasing feature size make I/O connection a critical problem. The flip-chip package is introduced for such designs with higher integration density and larger I/O counts. The flip-chip packaging is a technique for connecting a die to an external circuitry such as package carriers or printed circuit boards (PCB's). It can bring much more areas for I/O's, maintain high-speed performance with shorter interconnections, and keep signals isolated from environmental hazards or influences. As shown in Figure 1, a die is flipped over and mounted on a package carrier. To ease design changes, an extra metal layer, called a Redistribution Layer (RDL), is added and used to redistribute nets between the I/O pads and the bump pads [8], [9]. Accordingly, a flip-chip router is needed to connect the I/O pads to the bump pads. As the design complexity grows, in particular, routing has played a pivotal role in flip-chip design.

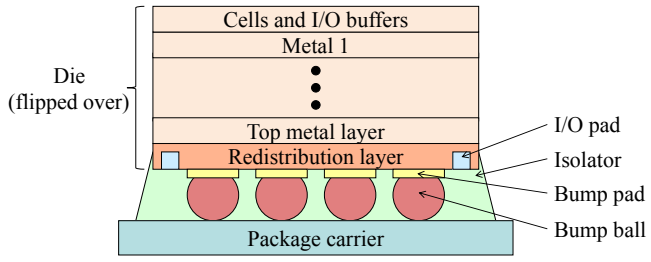

Figure 1. Cross view of a flip-chip.

The flip-chip package structure can be categorized into two major types: (1) the *peripheral-I/O* flip-chip [8], [9] and (2) the *area-I/O* flip-chip [3], [4], as illustrated in Figure 2. Generally speaking, the routing for the area-I/O flip-chip is harder than that for the peripheral-I/O one since the routing region among bump pads in an area-I/O flip-chip is more congested, as illustrated in Figure 2. On the other hand, the area-I/O flip-chip exhibits significant advantages in shorter connection wirelength, higher design flexibility, etc.

Depending on the interactions among IC, packaging, and PCB designers, the problem formulation of flip-chip RDL routing can be classified into three major types: (1) the *free-assignment* (FA for short) routing problem [3], [4], [8], [9], [12], [14], (2) the *pre-assignment* (PA) routing problem [6], [7], [11], [15], and (3) the *unified-assignment* (UA) routing problem [10]. For the FA problem, the net assignments between I/O pads and bump pads are not predefined before routing, so a router has the freedom to assign each I/O pad to an arbitrary bump pad. In contrast, for the pre-assignment problem, connections between I/O pads and bump pads are predefined by IC or packaging designers before routing, and the assignments cannot be changed during routing. The pre-assignment problem has been shown to be much more difficult than the free-assignment problem since the strict mapping between I/O pads and bump pads impose more design constraints [6], [7]. For the UA problem, some net assignments between I/O pads and bump pads are predefined, and some are not. As a result, both the FA and PA routing problems shall be considered simultaneously to achieve better design flexibility and performance [10].

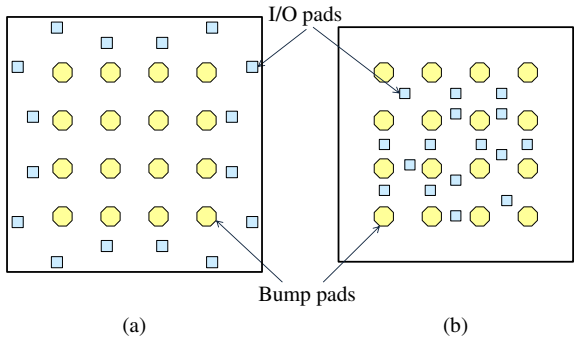

Figure 2. Flip-chip structures. (a) Peripheral-I/O flip-chip. (b) Area-I/O flip-chip.

Figure 3 classifies flip-chip routing problems with corresponding published research works, based on the flip-chip structures and pad assignment methods. In this paper, we first introduce the published research works and discuss their strengths and weaknesses, and then provide future research directions for modern flip-chip routing.

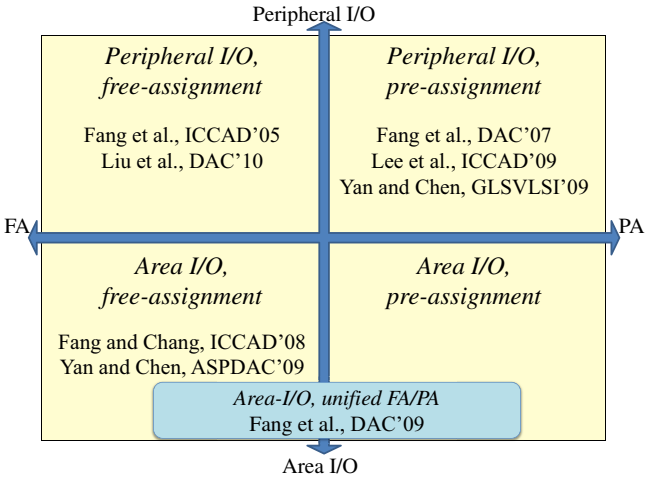

Figure 3. Classification of flip-chip routing problems and corresponding published works.

The rest of this paper is organized as follows. Sections II, III, and

978-1-4244-8192-7/10/$26.00 ©2010 IEEE

IV survey published works on the free-, pre-, and unified-assignment routing problems, respectively. Section V provides some future research directions for flip-chip routing, and the paper is concluded in Section VI.

II. The Free-Assignment Problem

In the free-assignment RDL routing problem, each I/O pad can be paired with any bump pad that is not connected with another net. To handle this problem, most published works utilize the network-flow-based algorithm in the global-routing stage [3], [4], [8], [9], [12], introduced in Subsection II-A. Another non-flow-based method [14] is introduced in Subsection II-B.

A. Network-Flow-Based Methods

The basic idea of network-flow-based methods is to model routing regions as a set of *tiles*, construct a flow network to model the original problem instance, apply the *Minimum-Cost Maximum-Flow* (MCMF) algorithm [1] on the corresponding flow network, and convert the resulting flow solution to a global-routing topology for the original problem. With the global-routing solution, a detail-routing algorithm is then applied to assign tracks and determine the actual position for each net.

Based on the routing-region modeling, we can further classify the network-flow-based methods into two major categories: (1) rectangular-tile model, and (2) triangular-tile model. We discuss published techniques for these two models in the following.

1) Rectangular-Tile Model:

a) Peripheral-I/O Structure: Fang *et al.* [8], [9] proposed the first work on flip-chip routing problem. A network-flow-based approach is used to handle the peripheral-I/O free-assignment RDL routing problem. Figure 4 illustrates an example chip instance, where bump pads and I/O pads are distributed *ring by ring*, with two I/O pad rings and three bump pad rings in the figure. The chip is divided into four sectors, namely North, East, South, and West, and then each sector is handled separately.

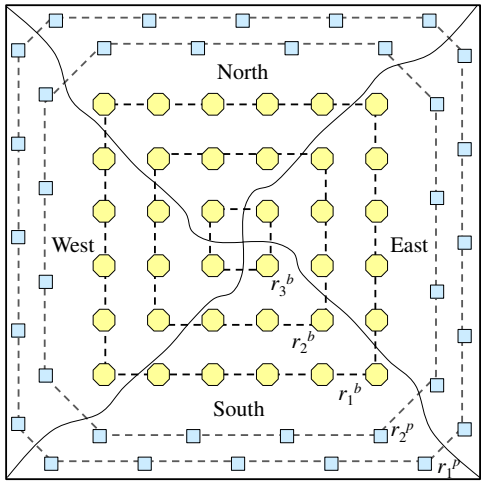

Figure 4. A peripheral-I/O flip-chip is divided into four sectors: North, East, South, and West. Two rings of I/O pads are denoted by r_1^p and r_2^p, and three rings of bump pads are denoted by r_1^b, r_2^b, and r_3^b, from the outer ring to the inner one.

They define a *tile* to be a rectangular region bounded by four bump pads and/or I/O pads, and an *interval* to be a segment between two adjacent bump pads or I/O pads on the same ring. Two kinds of nodes are constructed to control the congestion during network-flow processing: a *tile node* is constructed for each *tile* and an *intermediate node* is constructed for each *interval*. As shown in Figure 5(a), the upper four bump pads form a tile and two intervals, and so does the lower four I/O pads.

The flow edges are directional edges from the outer rings to the inner rings, since this work only consider *monotonic routes*, where no route passes through the same ring twice. Figure 5(b) is the flow network constructed for the entire South sector of the chip shown in Figure 4. The nodes s and t are the source and sink, respectively. Note that the sink t is connected from *all bump pads*; only three flow edges from bump pads to t are shown here for brevity.

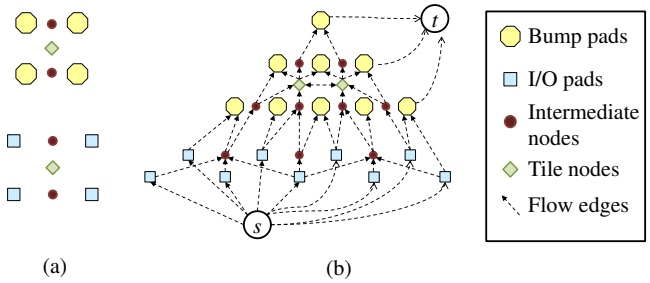

Figure 5. An example flow network construction. (a) Construction of tile and intermediate nodes. (b) The flow network for the *South* sector of the flip-chip shown in Figure 4. Nodes s and t are the source and sink, respectively. Note that the sink t is connected from *all bump pads*; only three edges from bump pads to t are shown here for brevity.

The congestion is controlled through the capacity constraint of tile and intermediate nodes. Each tile/intermediate node is assigned the capacity of the maximum number of nets that can pass through the corresponding tile/interval without violating design rules. The capacity of a tile (intermediate) node is denoted by C_t (C_d). The capacity of an edge is set as the maximum number of nets that can pass through in the edge's direction. For example, for 2-pin nets, the capacity of an edge connecting an I/O pad or a bump pad is set as 1, since only one net can be attached to a pad. And the capacity of an edge connecting an intermediate (tile) node is set as C_d (C_t).

The cost of an edge is set to be proportional to the Manhattan wirelength of this edge. Consequently, minimizing the total cost would lead to shorter total wirelength. With the capacity and cost settings, we can apply the MCMF algorithm on the resulting flow network to obtain global-routing topologies for all nets.

After global routing, detailed routing is performed to assign tracks and decide the exact location of each net. Detailed routing is not elaborated here, as global routing is more critical and thus we shall focus on it in this paper.

b) Area-I/O Structure: The area-I/O structure is also popular in real-world applications. Figure 6(a) illustrates an area-I/O flip-chip instance, where the I/O pads are located everywhere inside the bump-pad array, and thus the previous tile model is no longer applicable.

Fang and Chang [3], [4] proposed the first work on area-I/O free-assignment routing problem. They modified the flow network and the tile node capacity of the previous model in order to deal with the area-I/O structure. In the previous flow network, as shown in Figure 5(b), the directions of all edges are from outside to inside because all nets start from the outer I/O pad rings and end at inner bump pad rings. In the area-I/O structure, however, the nets can go any direction. So the flow edges are constructed in all directions, as shown in Figure 6(b). Moreover, the intermediate nodes are now constructed between every pair of adjacent bump pads to avoid congestion, while in the previous flow network the intermediate nodes are only constructed between pads of the same ring.

Another issue about the area-I/O structure is that the routing region of a tile is apparently reduced when I/O pads are inside the tile. Each I/O pad is punched through all redistribution layers, and thus the corresponding routing resource is occupied. To handle this issue, they also modified the capacity of a tile node to be the corresponding area proportion inside the tile; that is, $C_t' = \lfloor C_t \times (1-k) \rfloor$, where k is the area portion of the I/O pads in the tile.

2) Triangular-Tile Model: Fang *et al.* [10] first applied the Delaunay Triangulation (DT) and the Voronoi Diagram (VD) algorithm on

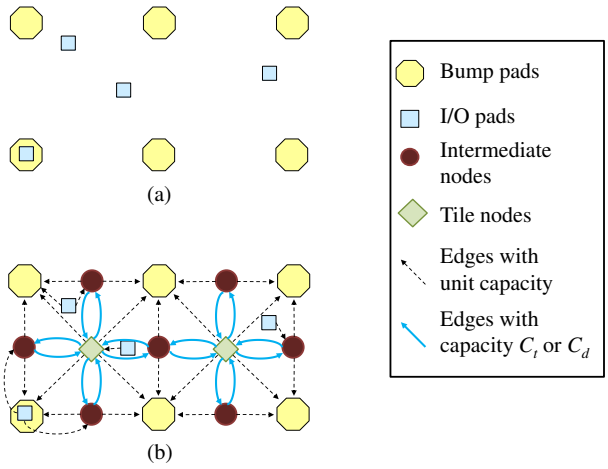

Figure 6. A sample area-I/O flip-chip instance. (a) The I/O pads lie within the bump-pad array. (b) The flow network is now composed of edges of all directions.

flip-chip routing. Their work focuses on the *unified-assignment* problem, and is introduced in Section IV. Note that although their work focuses on the UA routing problem, they still adopt a triangulation, network-flow-based algorithm to handle FA nets.

Liu *et al.* recently proposed a network-flow-based approach for the peripheral-I/O free-assignment RDL routing problem using the triangular-tile model. Their model is also based on DT and VD. Given a set of points P, Delaunay triangulation applies a computational geometry technique to triangulate the plane into triangular regions according to P. DT has a special property that no point lies inside the circumcircle of another triangle after DT. And the minimum angle of all the angles of the triangles is maximized. Also given $P = \{p_1, ...p_n\}$, Voronoi Diagram is a decomposition of a plane into Voronoi cells $C = \{c_1, ..., c_n\}$, that any point inside c_i is closer to p_i than any other $p_j \in P, i \neq j$. VD is the dual graph of DT and can directly be retrieved through DT.

They first computed the DT and VD of the set of points induced from the bump polygons, the centers of I/O pads, and some control points on the boundary of a chip. Figure 7(a) illustrates the corresponding DT and VD for a portion of a sample peripheral-I/O flip-chip. They then used VD as the flow network, and flow edges are constructed between VD nodes and pads. See Figure 7(b) for the resulting flow network. It is clear that the flow network formed by the VD edges well models the routing "channels" between pads.

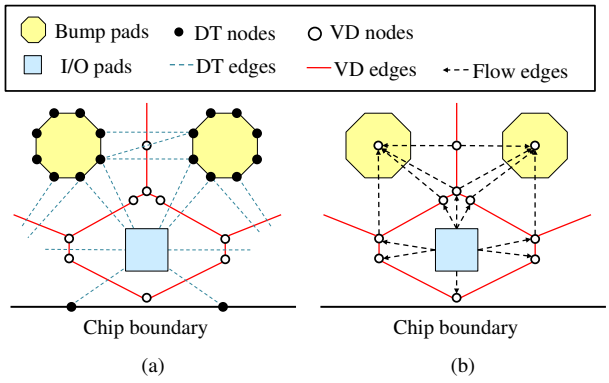

Figure 7. An example of using Delaunay triangulation on a peripheral-I/O flip-chip. (a) The Delaunay triangulation and Voronoi diagram of the flip-chip. (b) Flow network constructed according to the Voronoi diagram.

The cost and capacity setting of the flow network is similar to the previous models. The cost of an edge is given by its wirelength. And the capacity of edges connecting source/sink is set to 1 so that only one net can be attached to one bump or I/O pad. The cost of an VD edge is set to be proportional to the length of the corresponding edge in the dual graph DT. Generally speaking, each VD node represents the DT triangle it resides in, and the capacity of the edge connecting the VD node and its neighbor is set according to the length of the common edge of these two DT triangles.

In practice, the directions of nets are limited to 0-, 45-, 90-, and 135-degrees only. So in this work, the length of a DT edge is projected to four lines of 0-, 45-, 90-, and 135-degrees and take the maximum of the projection lengths to model the capacity of the corresponding VD edge. Further, some very short VD edges are removed both for efficiency and correctness, since those short edges are often improper routing channels with inaccurate capacity.

B. Non-Network-Flow-Based Methods

Yan and Chen presented a non-network-flow-based approach for area-I/O free-assignment RDL routing [14]. For the free-assignment RDL routing problem, it is most crucial to assign each I/O pad to a bump pad. While network-flow-based methods assign all pads concurrently through the MCMF algorithm, this work adopts a different approach of assigning I/O pads to bump pads.

The Voronoi diagram (VD) is applied to perform the assignment. The method first constructs the VD of all I/O pads, and acquires the Voronoi cell for each I/O pad. By the property of VD, each cell contains exactly one I/O pad, but the number of bump pads inside the cell varies. If an I/O pad's cell contains at least one bump pad, the I/O pad is assigned to the closest bump pad in the cell. Otherwise, the I/O pad remains unassigned until the next iteration. Figure 8(a) illustrates a sample area-I/O flip-chip instance. After the VD is constructed, the I/O pads are assigned to the nearest bump pads in the cell, and only one I/O pad remains unassigned after the first iteration.

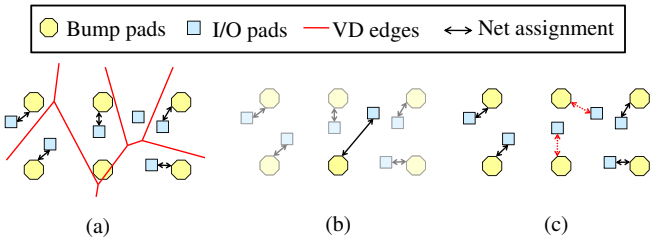

Figure 8. The pad-assignment procedure of Yan and Chen's work. (a) After the VD of I/O pads is constructed, each I/O pad is assigned to the nearest bump pad in the VD cell. (b) The assigned pads are ignored, and the remaining pads are repeatedly assigned by the same method. (c) After the initial assignment, some pairs of assignments are exchanged (the dashed assignments) to reduce the total wirelength and/or increase routability.

After each iteration, the assigned pads are ignored, and only unassigned pads are considered during the next VD generation. As shown in Figure 8(b), the unassigned pad is assigned in the second iteration.

This procedure generates the initial assignment. Next, it adjusts the initial assignment by exchanging assignment pairs to reduce wirelength and removes crossing pairs. For example, the assignments of the two I/O pads in the center of Figure 8(b) can be exchanged to achieve shorter wirelength. The swapped assignments are denoted by dashed lines in Figure 8(c).

After the pad assignment, the global-routing topology is decided for each net by a global wire assignment step. Detailed routing is then carried out by river routing followed by maze routing.

C. Remarks

The network-flow-based formulation is very popular for handling the free-assignment RDL routing problem because it can determine the global routes concurrently. More importantly, the flow running freely in a

network well matches the intrinsic nature of *free* assignment of nets in a flip-chip. The most essential part of a network-flow-based algorithm lies in a good tile model that the capacity of nodes and edges can accurately represent the routing resource so that the design rules are not violated.

For the two tile models, the rectangular-tile model is naturally formed with the regular structure of the bump-pad array, while the triangular-tile model is formed with Delaunay triangulation. Generally speaking, the triangular-tile model is more flexible for various flip-chip structures, but it is less accurate in capacity modeling for layouts corresponding to irregular triangles. The strengths and weaknesses of the two tile models are further discussed in Section V.

III. Pre-Assignment Problem

For the pre-assignment problem, each I/O pad is assigned to a bump pad according to a pre-defined netlist. Published approaches for this problem can be classified into two categories: (1) ILP-based methods and (2) non-ILP-based methods.

A. ILP-Based Method

For the pre-assignment problem, the network-flow-based methods are no longer applicable because each I/O pad must be connected to a predefined bump pad, which violates the nature of flows in a network. Under the pre-assignment constraint, the problem becomes a *multi-commodity flow* problem [1], which is known to be NP-complete for integral flows. In this subsection, we introduce an *Integer Linear Programming* (ILP) formulation to solve this problem.

Fang, Hsu, and Chang [6], [7] first address the peripheral-I/O pre-assignment problem. Their global-routing algorithm consists of two parts: (1) *ILP network construction* and (2) *ILP formulation*.

1) ILP Network Construction: A peripheral-I/O pre-assignment flip-chip has the same pad structure as that shown in Figure 4, except that each I/O pad is now assigned to a specific bump pad.

Similar to the free-assignment work in Section II-A1, an *interval* is defined to be the segment between two adjacent bump pads or two I/O pads, and a *tile* to be the rectangular region bounded by four adjacent pads. However, here *ILP nodes* are constructed, instead of tile and intermediate nodes.

Figure 9(a) illustrates an example with three nets, nets 1, 2, and 3. Three I/O pads are assigned to two bump pads since nets 1 and 2 are assigned to the same bump pad.

ILP nodes are constructed in a way that the nodes of outer rings are propagated to inner rings, ring by ring. Figure 9(b) illustrates the ILP network of nets 1. 2. and 3. The procedure of generating the network in Figure 9(b) begins at the outermost ring, r_1^p. No ILP nodes are constructed in this ring since no other nets could possibly pass through this ring. Then, the configuration of r_1^p must be propagated to the next ring, r_2^p. There are three possible locations for net 1 to pass through r_2^p: (1) to the left of I/O pad 2, (2) between I/O pads 2 and 3, and (3) to the right of I/O pad 3. Therefore, three ILP nodes representing net 1 are constructed at these locations. As a result, r_2^p now consists of five nodes: $< 1, 2, 1, 3, 1 >$, as shown in Figure 9(b).

Next, it propagates the nodes on r_2^p to the next ring. Similarly, each of the five nodes in r_2^p can pass through r_1^b through three different locations. So totally 15 ILP nodes are constructed for r_1^b.

Finally, propagating the nodes of r_1^b to r_2^b, we find that all the three nets are assigned to bump pads in this ring. Therefore, each net has exactly one possible location in this ring—the assigned bump pad. No additional ILP nodes are constructed in this ring, and the network in Figure 9(b) is thus successfully generated.

2) ILP Formulation: After constructing the ILP network, we now need to express the objective function and constraints in the ILP form.

We need the following notations for the ILP formulation:

- $x_{i,j}$: 0-1 integer variable that denotes if a candidate segment j is chosen in the global-routing path of net n_i. $x_{i,j}$=1 if the segment j is chosen; $x_{i,j}$=0, otherwise.
- $e_{i,j}$: edge that denotes a candidate segment j of the global-routing path of n_i.
- $L(e_{i,j})$: function that denotes the length of $e_{i,j}$.

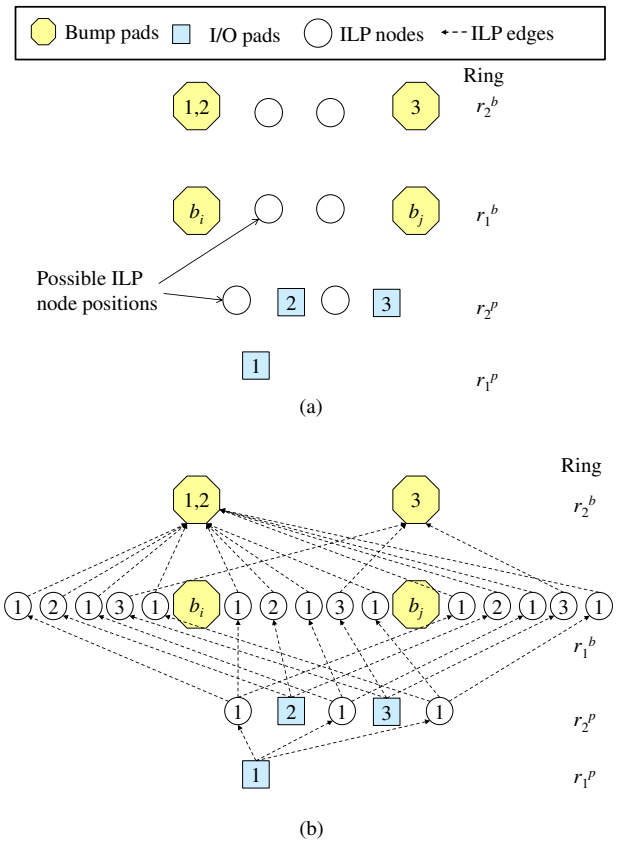

Figure 9. ILP node construction. (a) ILP nodes are constructed for an interval. (b) The complete ILP network for nets 1, 2, and 3.

- $W(e_{i,j})$: function that denotes the wire width of net n_i.
- $C(e_{i,j}, e_{p,q})$: function that denotes the crossing between $e_{i,j}$ and $e_{p,q}$. If $e_{i,j}$ crosses $e_{p,q}$, $C(e_{i,j}, e_{p,q})$=1; otherwise, $C(e_{i,j}, e_{p,q})$=0.
- $P_i(e_{i,j})$: function that denotes the connection of $e_{i,j}$ and I/O pad $p_i \in P$. If $e_{i,j}$ connects p_i, $P_i(e_{i,j})$=1; otherwise, $P_i(e_{i,j})$=0.
- $D_k^{in}(e_{i,j})$: function that denotes the connection of $e_{i,j}$ and the input side of ILP node $d_k \in D$. If $e_{i,j}$ connects the input side of d_k, $D_k^{in}(e_{i,j})$=1; otherwise, $D_k^{in}(e_{i,j})$=0.
- $D_k^{out}(e_{i,j})$: function that denotes the connection of $e_{i,j}$ and the output side of ILP node $d_k \in D$. If $e_{i,j}$ connects the output side of d_k, $D_k^{out}(e_{i,j})$=1; otherwise, $D_k^{out}(e_{i,j})$=0.
- $T_m(e_{i,j})$: function that denotes the existence of $e_{i,j}$ in tile $m \in M$. If $e_{i,j}$ is in tile m, $T_m(e_{i,j})$=1; otherwise, $T_m(e_{i,j})$=0.
- t_m: constant that denotes the routing resource of tile $m \in M$.
- $H_u(e_{i,j})$: function that denotes the existence of $e_{i,j}$ in interval $u \in U$. If $e_{i,j}$ is in interval u, $H_u(e_{i,j})$=1; otherwise, $H_u(e_{i,j})$=0.
- h_u: constant that denotes the routing resource of interval $u \in U$.
- $s_{i,p}$: constant that denotes the maximum allowance of the signal skew between net i and net p. Each $s_{i,p}$ is in the constraints F.

With these notations, the RDL routing problem can be formulated as follows:

$$\min \sum_{e_{i,j} \in E} L(e_{i,j}) x_{i,j}$$

subject to

$$C(e_{i,j}, e_{p,q})(x_{i,j} + x_{p,q}) \leq 1, \forall e_{i,j}, e_{p,q} \in E, \quad (1)$$

$$\sum_{e_{i,j} \in E} W(e_{i,j}) T_m(e_{i,j}) x_{i,j} \leq t_m, \forall m \in M, \quad (2)$$

$$\sum_{e_{i,j} \in E} W(e_{i,j}) H_u(e_{i,j}) x_{i,j} \leq h_u, \forall u \in U, \quad (3)$$

$$\left| \sum_{j \in n_i} L(e_{i,j}) x_{i,j} - \sum_{q \in n_p} L(e_{p,q}) x_{p,q} \right| \leq s_{i,p}, \forall s_{i,p} \in F, \quad (4)$$

$$\sum_{e_{i,j} \in E} P_i(e_{i,j}) x_{i,j} = 1, \forall p_i \in P, \quad (5)$$

$$\sum_{e_{i,j} \in E} D_k^{out}(e_{i,j}) x_{i,j} = \sum_{e_{i,q} \in E} D_k^{in}(e_{i,q}) x_{i,q}, \forall d_k \in D, \quad (6)$$

- The objective function is to minimize the total wirelength under the constraints.
- Constraint (1) avoids the crossing between nets: if two edges cross each other, at most one can exist.
- Constraint (2) is used to avoid the congestion overflow of a tile since there may be too many edges passing through the *tile* formed by four bump pads.
- Constraint (3) is used to avoid the congestion overflow of an *interval* between two pads, similarly.
- Constraint (4) formulates the signal-skew constraint between two nets, which is not of our main concerns here.
- Constraint (5) guarantees that at least one edge of the I/O pad p_i of net n_i be chosen, that is, every net be routed.
- Constraint (6) is for the flow conservation; that is, the total flow of the output side and the input side must be the same.

After solving this ILP formulation, global routing is completed, and then detailed routing is applied to compete the routing.

Note that this ILP formulation is actually a modeling of the *multi-commodity flow* problem. Constraint (5) assigns unit flow to each I/O pad. Constraint (6) ensures flow conservation for each node. Constraints (2) and (3) control congestion of tiles and intervals, just like the capacity of tile and intermediate nodes in a free-assignment flow network. Additionally, Constraint (1) forbids net crossing, and Constraint (4) limits the signal skew.

B. Non-ILP-Based Methods

Unlike the ILP-based methods, some other works take advantage of the regular structure of a flip-chip to develop more efficient algorithms to solve the problem.

Lee *et al.* in [11] developed an algorithm based on *net sequence exchange* for pad rings. The *net sequence* of a pad ring is defined as the order of nets going outward from the ring. As shown in Figure 10(a), the sequence of the upper I/O pad ring is $< 1, 2 >$, and the sequence of lower bump pad ring is $< 2, 1 >$. If the pads are connected directly, there will be a crossing between nets 1 and 2. To avoid crossing, we must either exchange the I/O pad ring sequence to $< 2, 1 >$ and detour net 2 (Figure 10(b)), or exchange the bump pad ring sequence to $< 1, 2 >$ and detour net 1 (Figure 10(c)). They observed that the number of detours should be minimized, since detours consume more routing resources and decrease routability.

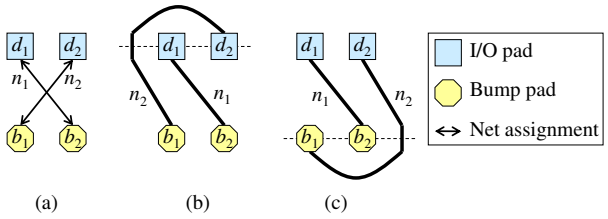

Figure 10. Example of sequence exchange. (a) If I/O pads and bump pads are connected directly, there will be a crossing. Therefore, the net sequence must be exchanged (b) on the I/O-pad side or (c) on the bump-pad side to complete the routing.

Based on this concept, it iteratively routes bump pads outwards, from the innermost bump pad ring to the outermost bump pad ring. In each iteration, two dynamic programming algorithms are applied to exchange the sequence: (1) weighted longest common subsequence (Weighted LCS) [2] and (2) maximum planar subset of chords (MPSC) [13]. Both algorithms are applied to maximize the number of direct routes and minimize the number of detoured nets.

An example of applying the LCS algorithm to minimize detoured nets is shown in Figure 11(a): There are three I/O pads, d_1, d_2, d_3 and three bump pads, b_1, b_2, b_3. d_i is assigned to b_i, $i = 1, 2$, and 3. The input of the LCS computation is the two sequences, $< 1, 2, 3 >$ and $< 3, 1, 2 >$. There are two maximal common subsequences, $< 3 >$ and $< 1, 2 >$; each subsequence represents a set of nets that can be directly routed without net crossing. The resulting topologies of directly routing $< 3 >$ and $< 1, 2 >$ are shown in Figure 11(b) and Figure 11(c), respectively. It is clear that the result of routing $< 1, 2 >$ directly is indeed better, implying the preference for a longer LCS. In the original paper, the LCS algorithm is extended to weighted LCS to further minimize wirelength (weight).

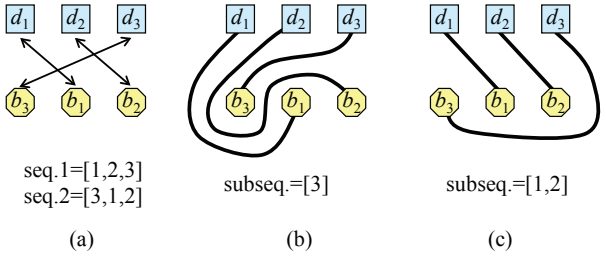

Figure 11. An example of the LCS computation. (a) Two sequences. (b) One possible subsequence. (c) The other possible subsequence.

The nets not in the LCS must be detoured. These detoured nets might cross with nets coming from the inner rings. The MPSC computation is used to determine which nets should be routed directly, and which nets should be further detoured. For example, as shown in Figure 12(a), net 3 is forced to be detoured in the previous LCS computation, and cross with nets 4 and 5 coming from inner rings. To decide which nets should be detoured further, a circle is constructed to enclose these three nets, and each net is regarded as a chord. Then, the MPSC algorithm is applied to get the maximum set of non-crossing chords (nets). In this example, the MPSC is $\{4, 5\}$. Figures 12(b) and (c) illustrate that the topology of directly routing nets $\{4, 5\}$ and further detouring net $\{3\}$ (Figure 12(c)) is better than the opposite (Figure 12(b)).

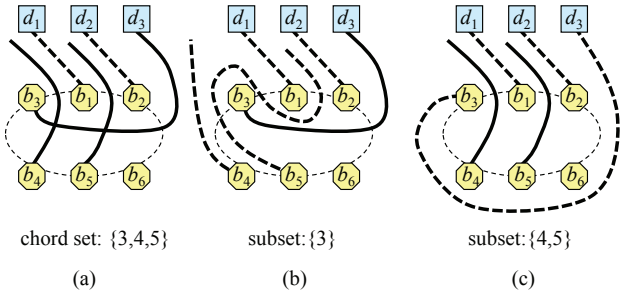

Figure 12. An example of the MPSC computation. (a) The circle and the set of chords (solid lines). (b) One possible subset. (c) A better subset with two non-crossing chords.

Their experiments show that the algorithm not only has a huge speedup over the previous ILP-based method, but also achieves better quality (due to some inaccurate tile modeling in the ILP-based method). See Section V for further discussions.

Yan and Chen [15] also proposed a pre-assignment routing algorithm for peripheral-I/O flip-chip. Their work is based on a similar concept of maximizing direct routes and minimizing detoured nets.

C. Remarks

ILP is a flexible method for solving the general flip-chip routing problem. However, ILP by its nature is a brute-force approach that basically tries every possible solution blindly. While it guarantees to find the "optimal" solution in the solution space, it is typically very time-consuming.

On the other hand, both the non-ILP-based methods observe and utilize the regularity of the peripheral-I/O structure to develop more efficient algorithms to minimize the number of detoured nets, thus improving the routing result.

IV. Unified-Assignment Problem

In modern designs, a routing instance could contain both free- and pre-assignment nets. This is mostly caused by the designers pre-defining the pin assignments for some crucial nets such as power and clock signals, while leaving other less important nets unassigned. As a result, it is desirable to develop a routing algorithm that can consider PA and FA nets simultaneously, instead of handling PA and FA nets separately.

A. Triangular-Tile Model

Fang, Wong, and Chang proposed the first work for the area-I/O unified-assignment (UA) RDL routing problem. As mentioned in Section II, their method is actually the first work utilizing Delaunay triangulation and the Voronoi diagram to handle flip-chip routing. Figure 13(a) illustrates an area-I/O UA flip-chip instance, and the DT and VD results. Note that all I/O pads are labeled with net ID's, while only some of the bump pads are labeled with net ID's. An I/O pad whose net ID is also labeled on a bump pad is a PA net, and must be connected to the bump pad with the same ID. On the other hand, an I/O pad whose net ID does not appear on any bump pad is a FA net, and can be connected to any unlabeled bump pad.

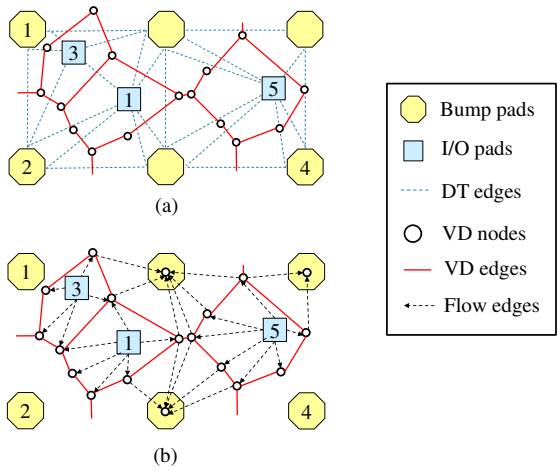

Figure 13. An example of area-I/O UA routing. Note that all I/O pads are labeled with net ID's, while only some of the bump pads are labeled with net ID's. (a) The Delaunay triangulation and Voronoi diagram of the flip-chip, similar to Figure 6. (b) Flow network constructed according to the Voronoi diagram.

After computing the VD as shown in Figure 13(a), we can get the flow network by adding edges connecting VD nodes and nearby pads. Figure 13(b) gives the resulting flow network. Note that this example is the FA flow network, so only FA bump pads are connected. The PA bump pads are connected to the flow network when we need to route the PA nets.

The algorithm consists of four stages: (a) congestion estimation, (b) PA nets routing, (c) FA nets routing, and (d) iterative improvement.

a) Congestion Estimation: Congestion estimation is performed by simply applying the MCMF algorithm on the FA flow network, like the network in Figure 13(b). In this stage, the FA nets are *not* actually routed. However, the potentially congested nodes and edges are identified.

b) PA Net Routing: The PA bump pads are first connected to the flow network, then the PA nets are routed using maze routing. The costs of nodes and edges, previously used by FA nets, are now raised much higher, implying a penalty to potential congestion and wire crossing between FA and PA nets.

c) FA Net Routing: After the PA nets are routed, we can now actually route the FA nets. The previously routed PA nets are treated as obstacles, and edges crossing these PA nets are assigned high costs to prevent wire crossing. Again, the MCMF algorithm is applied to get the routing results of FA nets.

d) Iterative Improvement: After the FA routing, there may be some wire crossings between FA and PA nets. The reasons are that there is no solution without wire crossings or the detours of the FA nets are too long. Then we rip up and reroute the PA nets crossing the routed FA wires. We also rip up and reroute the FA nets if the cost of rerouting the PA nets is too high, or simply impossible. Finally, the iterative improvement stage converges.

Their experiments show that considering PA and FA nets simultaneously achieves better results than considering them separately.

B. Remarks

This work provides a method to handle the area-I/O UA RDL routing problem. It shows that the triangular-tile model can be used to handle both PA and FA nets. Moreover, the FA part of the algorithm itself can be extracted as an area-I/O FA routing algorithm. Their experiments also show that the co-consideration of different issues are crucial when facing problems consisting of multiple issues.

V. Future Research Directions

Although recent works have made significant progress in the flip-chip routing problem, there are still many emerging challenges arising from advanced technologies and designs. In this section, we present some potential research directions for modern flip-chip routing.

A. Tile Modeling

Constructing a tile model that accurately represents the capacity of the corresponding routing region is crucial for all network-flow and multi-commodity-flow (ILP) based methods. However, most published works still suffer from the accuracy problem in their modeling.

All published works set the capacity of each tile to be a single integer (constant), denoting the maximum number of nets that can pass through the tile without violating design rules. In practice, however, the number of nets that can pass through a tile is affected by the routing configuration inside the tile. Figure 14 illustrates how the capacity of a tile varies with different routing configurations.

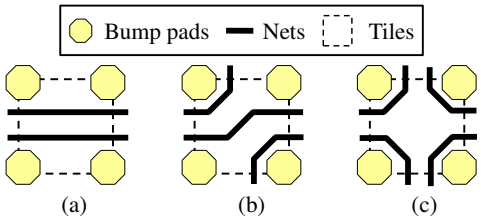

Figure 14. Capacity of a tile varies with different routing configurations. (a) Capacity is 2. (b) Capacity is 3. (c) Capacity is 4.

We also observe that the triangular-tile model is less accurate in capacity estimation than the rectangular-tile one, since the boundary of a triangular tile can be of any direction. This situation is even worse when the triangle is slim and/or tilted.

Yan and Wong proposed a rectangular-tile model that can accurately model the routing capacity for the escape routing problem [16]. This model can be applied to flip-chip routing. However, it can only model square tiles and thus cannot readily be used for area-I/O flip-chips. Further, it does not handle obstacles and irregular structures to be discussed later.

Generally speaking, the rectangular-tile model is more accurate than the triangular-tile one in terms of net capacity. In contrast, the triangular-tile model is more flexible and is not limited to a rectangular pad array structure. It is thus desirable to develop a model that is both accurate and flexible.

B. Obstacle Handling

Obstacles are unavoidable in flip-chip routing because of pre-placed modules and/or pre-routed nets, as shown in Figure 15. In presence of obstacles, the traditional rectangular-tile model may not be accurate since the tiles might partially be blocked or cut in half. On the other hand, the triangular-tile model can still be constructed by triangulating the chip plane. As discussed earlier, however, the triangular-tile model suffers from the accuracy problem in capacity estimation.

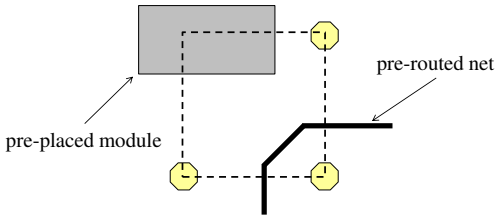

Figure 15. An example of pre-placed modules and pre-routed nets.

C. Irregular Structure Handling

In modern designs, the bump-pad array on a flip-chip is not always regular. Figure 16 illustrates two kinds of irregular structures in practical designs. Figure 16(a) shows a staggered bump-pad array, where the inner bump pads are literally "staggered". Figure 16(b) shows a special structure that the inner I/O pad ring locates inside the bump-pad array.

For these irregularly structured designs, it is desirable to develop a flexible algorithm that can handle not only known regular structures, but also other irregular structures yet to come. We believe that a better way to deal with these structures is to find the *regularity* within these *irregular structures*, and apply algorithms based on simple, yet essential concepts, like the pre-assignment non-ILP-based method [11] surveyed in Section III-B.

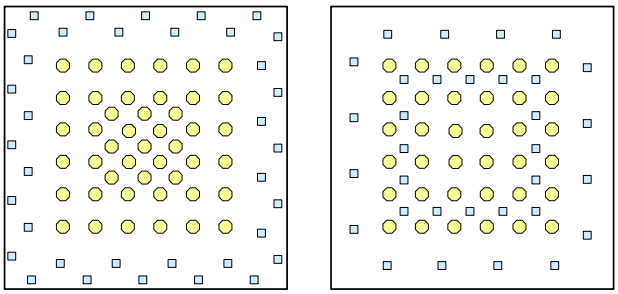

Figure 16. (a) Staggered bump-pad array. (b) Inner I/O pad ring structure.

D. Chip-Package-Board Co-Design

Advanced technologies have complicated the designs of chips as well as packages and PCB's. In order to improve the deign performance and convergence among them, chip-package-board co-design is strongly recommended by industry. Fang, Ho, and Chang published an ILP-based routing algorithm for chip-package-board co-design [5]. Although the algorithm guarantees to find an optimal solution, as expected, this algorithm is time-consuming. It is thus desirable to explore the special properties of chip-package-board co-design to develop more efficient algorithms for this problem.

E. Electrical Effect Optimization

For high-speed applications, many electrical effects need to be considered. For example, it is important to minimize the difference of the path lengths between two nets (i.e., signal skew), to minimize the coupling length between two nets with different signal polarity (i.e., signal integrity), etc. Many more electrical effects need to be considered in flip-chip routing for more advanced technologies yet to come.

VI. Conclusions

Routing is a critical step in modern flip-chip design. In this paper, we have surveyed published techniques to tackle the flip-chip routing problems with various combinations of flip-chip structures (peripheral-I/O and area-I/O structures), tile modeling (rectangular and triangular tiles), and pad assignments (free-, pre-, and unified-assignment between I/O pads and bump pads). Common techniques (such as the network-flow-based method and the ILP-based method) for generic structures as well as ad hoc techniques (such as dynamic programming and greedy heuristics) for particular structures are compared. Although significant progress has been made in flip-chip routing research, modern flip-chip designs need to consider more complex constraints and flexible structures, inducing many more challenges and opportunities for future research on tile modeling, obstacle handling, irregular flip-chip structure handling, chip-package-board co-design, and electrical effect optimization for the modern flip-chip routing problem.

VII. Acknowledgements

This work was partially supported by SpringSoft, Synopsys, TSMC, and NSC of Taiwan under Grant No's. NSC 98-2622-E-002-005-A2, NSC 98-2221-E-002-119-MY3, NSC 97-2221-E-002-237-MY3, NSC 96-2628-E-002-249-MY3, and NSC 96-2628-E-002-248-MY3.

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*, 2nd edition, The MIT Press, 2001.
[3] J.-W. Fang and Y.-W. Chang, "Area-I/O flip-chip routing for chip-package co-design," *Proc. ICCAD*, pp. 518–522, 2008.
[4] J.-W. Fang and Y.-W. Chang, "Area-I/O flip-chip routing for chip-package co-design considering signal skews," *IEEE Trans. on Computer-Aided Design*, vol. 29, pp. 711–721, 2010.
[5] J.-W. Fang, K.-H. Ho, and Y.-W. Chang, "Routing for chip-package-board co-design considering differential pairs," *Proc. ICCAD*, 2008.
[6] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang, "An integer linear programming based routing algorithm for flip-chip design," *Proc. of DAC*, pp. 606–611, 2007.
[7] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang, "An integer linear programming based routing algorithm for flip-chip designs," *IEEE Trans. Computer-Aided Design*, Vol. 28, No. 1, pp. 98–110, 2009.
[8] J.-W. Fang, I.-J. Lin, Y.-W. Chang, and J.-H. Wang, "A network-flow based RDL routing algorithm for flip-chip design," *IEEE Trans. on Computer-Aided Design*, vol. 26, pp. no. 8, pp. 1417–1429, 2007.
[9] J.-W. Fang, I-J. Lin, P.-H. Yuh, Y.-W. Chang, and J.-H. Wang, "A routing algorithm for flip-chip design," *Proc. ICCAD*, pp. 753–758, 2005.
[10] J.-W. Fang, M.D.F. Wong, and Y.-W. Chang, "Flip-chip routing with unified area-I/O pad assignments for package-board co-design," *Proc. DAC*, pp. 336–339, 2009.
[11] P.-W. Lee, C.-W. Lin, and Y.-W. Chang, "An efficient pre-assignment routing algorithm for flip-chip designs," *Proc. ICCAD*, pp. 239–244, 2009.
[12] X. Liu, Y. Zhang, G. K. Yeap, C. Chu, J. Sun, and X. Zeng, "Global routing and track assignment for flip-chip designs," *Proc. DAC,* pp. 90–93, 2010.
[13] K. J. Supowit, "Finding a maximum planar subset of a set of nets in a channel," *IEEE Trans. on Computer-Aided Design*, vol. 6, no. 1, pp. 93–94, 1987.
[14] J.-T. Yan and Z.-W. Chen, "IO connection assignment and RDL routing for flip-chip designs," *Proc. ASP-DAC*. pp. 588–593, 2009.
[15] J.-T. Yan and Z.-W. Chen, "RDL pre-assignment routing for flip-chip designs," *Proc. GLSVLSI*, pp. 401–404, 2009.
[16] T. Yan and M. D.-F. Wong, "A correct network flow model for escape routing," *Proc. DAC*, pp. 332–335, 2009.