A Network-Flow-Based RDL Routing Algorithmz for Flip-Chip Design

Jia-Wei Fang, *Student Member, IEEE*, I-Jye Lin, *Student Member, IEEE*, Yao-Wen Chang, *Member, IEEE*, and Jyh-Herng Wang, *Member, IEEE*

***Abstract*—The flip-chip package gives the highest chip density of any packaging method to support the pad-limited application-specific integrated circuit designs. In this paper, we propose the *first* router for the flip-chip package in the literature. The router can redistribute nets from wire-bonding pads to bump pads and then route each of them. The router adopts a two-stage technique of global routing followed by detailed routing. In global routing, we use the network flow algorithm to solve the assignment problem from the wire-bonding pads to the bump pads and then create the global path for each net. The detailed routing consists of three stages, namely: 1) cross-point assignment; 2) net ordering determination; and 3) track assignment, to complete the routing. Experimental results based on seven real designs from the industry demonstrate that the router can reduce the total wirelength by 10.2%, the critical wirelength by 13.4%, and the signal skews by 13.9%, as compared with a heuristic algorithm currently used in industry.**

***Index Terms*—Detailed routing, global routing, physical design.**

I. Introduction

A. Flip-Chip Design

DUE TO THE increasing complexity and decreasing feature size of very large scale integration (VLSI) designs, the demand of more I/O pads has become a significant problem of package technologies. A relatively new packaging technology, i.e., the *flip-chip package*, as shown in Fig. 1, is created for higher integration density and rising power consumption. Flip-chip bonding was first developed by IBM in the 1960s. It gives the highest chip density of any packaging method to support the pad-limited application-specific integrated circuit designs.

Flip-chip is not a specific package, or even a package type, e.g., pin grid array (PGA) or ball grid array (BGA). Flip-chip describes the method of electrically connecting the die to the package carrier. The package carrier, which is either a substrate or a lead frame, provides the connection from the die to the outside devices of the package. The die of a PGA/BGA package is attached to the carrier face up, and later, a wire is bonded first to the die, then looped and bonded to the carrier. In contrast, the interconnection between the die and carrier in the flip-chip package is made through a conductive bump ball that is placed directly on the die surface. Finally, the bumped die is flipped over and placed face down, with the bump balls connecting to the carrier directly. The flip-chip technology is the choice in high-speed applications because of the following advantages: reduced signal inductance (high speed), reduced power/ground inductance (low power), reduced package footprint, smaller die size, higher signal density, and lower thermal effect. However, in recent integrated circuit designs, the I/O pads are still placed along the boundary of the die. This placement does not suit the flip-chip package. As a result, we use the top metal or an extra metal layer, which is called a *redistribution layer* (RDL), as shown in Fig. 2, to redistribute the *wire-bonding pads* to the *bump pads* without changing the placement of the I/O pads. Since the RDL is the top metal layer of the die, the routing angle in an RDL cannot be any angle as in the PGA/BGA packages. Bump balls are placed on the RDL and use the RDL to connect to wire-bonding pads by bump pads.

The flip-chip package is generally classified into two types, namely: 1) the *peripheral array*, as shown in Fig. 3(a), and 2) the *area array*, as shown in Fig. 3(b). In the peripheral array, the bump balls are placed along the boundary of the flip-chip package. The disadvantage of the peripheral array is that we only have a limited number of bump balls. In the area array,

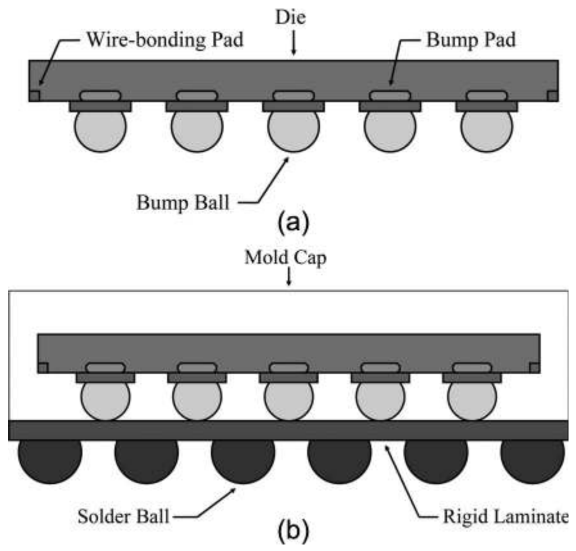


Fig. 1. (a) Flip-chip. (b) Flip-chip package.

Manuscript received January 4, 2006; revised September 17, 2006. This work was supported in part by Faraday Technology Corporation of Taiwan and in part by the National Science Council of Taiwan under Grants NSC 93-2215-E-002-009, NSC 93-2215-E-002-029, and NSC 93-2752-E-002-008-PAE. This paper was presented in part at the 2005 IEEE/ACM International Conference on Computer-Aided Design, San Jose, CA, November 2005. This paper was recommended by Associate Editor J. Hu.

J.-W. Fang and I-J. Lin are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: jiawei@eda.ee.ntu.edu.tw; waves@eda.ee.ntu.edu.tw).

Y.-W. Chang is with the Department of Electrical Engineering and Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan, R.O.C. (e-mail: ywchang@cc.ee.ntu.edu.tw).

J.-H. Wang is with Apache Design Solutions, Hsinchu 300, Taiwan, R.O.C. (e-mail : jhwang@apache-da.com)

Digital Object Identifier 10.1109/TCAD.2007.891364

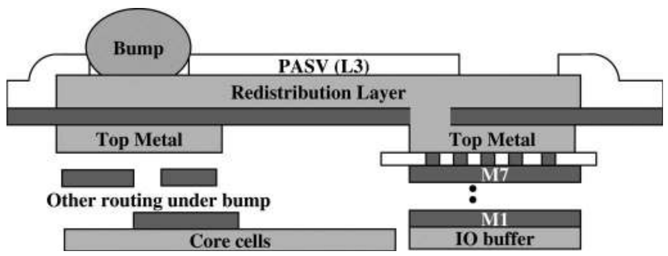


Fig. 2. Cross section of RDL.

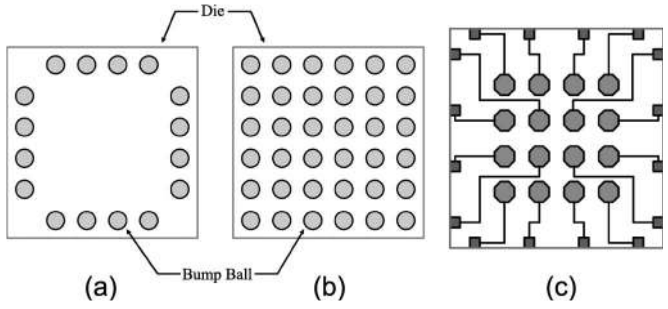


Fig. 3. (a) Peripheral array. (b) Area array. (c) RDL routing result.

the bump balls are placed in the whole area of the flip-chip package. The advantage of the area array is that the number of bump balls is much more than that of the peripheral array; thus, it is more suitable for modern VLSI designs. Since the flip-chip design is for high-speed circuits, the issue of signal skews is also important. Thus, a special router, i.e., the *RDL router* [11], is needed to reroute the peripheral wire-bonding pads to the bump pads and then connect the bump pads to the bump balls. Consider that the routing of multipin nets and the minimization of the total wirelength and the signal skews are also needed for an RDL router. Fig. 3(c) shows one RDL routing result for an area-array flip-chip.

B. Previous Work

To the best knowledge of the authors, there is no previous work in the literature on the routing problem for flip-chip designs. Similar works are the routing for PGA packages, BGA packages, and planar graphs, including [1]–[4], [8]–[10], and [12]–[15]. Yu and Dai [14] used the geometric and symmetric attributes of the pin positions in the BGA packages to assign pins of the BGA packages. However, in flip-chip designs, the positions of wire-bonding pads and bump pads do not always have these geometric and symmetric attributes. PGA routers are presented in [3] and [10], whereas a BGA router is provided in [4]. These three routers are any-angle multilayer routers without considering the pin assignment problem, single-layer routing, and total wirelength minimization. Wang *et al.* [12] and Yu *et al.* [15] applied the minimum-cost network flow algorithm to solve the I/O pin routing problems. All these routers focused only on routability and did not consider multipin nets and signal skews. Wang *et al.* [12] also did not consider the routing congestion problem. Furthermore, they assumed that wires can be any angle; thus, their methods are not suitable for the RDL routing, typically with a 90° angle routing. For the previous works on the planar routing [1], [2], and [5], since the pins can be placed anywhere in the chip, it is a nameplate-complete problem, and thus, most likely, there exists no efficient optimal algorithm for the planar routing. In the flip-chip routing, since wire-bonding pads and bump pads are placed in arrays, we can take the advantage of the regular structure to find an efficient algorithm for the RDL routing. Thus, the flip-chip routing problem is also different from the planar routing one.

C. Our Contributions

To our best knowledge, this paper is the first work in the literature to propose an RDL router to handle the routing problem of flip-chip designs with real industry applications. We present a unified network flow formulation to simultaneously consider the concurrent assignment of the wire-bonding pads to the bump pads and the routing between them. Our algorithm consists of two phases. The first phase is the global routing that assigns each wire-bonding pad to a unique bump pad. By formulating the assignment as a maximum-flow problem and applying the minimum-cost maximum-flow (MCMF) algorithm, we can guarantee 100% detailed routing completion after the assignment. The second phase is the detailed routing that efficiently distributes the routing points between two adjacent wire-bonding (bump) pads and assigns wires into tracks. In addition to the traditional single-layer routing with only routability optimization, our RDL router also tries to optimize the total wirelength and the signal skews between a pair of signal nets under the 100% routing completion constraint. Experimental results based on seven real designs from the industry demonstrate that the router can reduce the total wirelength by 10.2%, the critical wirelength by 13.4%, and the signal skews by 13.9%, as compared with a heuristic algorithm currently used in industry.

The rest of this paper is organized as follows: Section II gives the formulation of the RDL routing problem. Section III details our global and detailed routing algorithms. Section IV shows the experimental results. Finally, conclusions are given in Section V.

II. Problem Formulation

We introduce the notations used in this paper and formally define the routing problem for flip-chip packages. Fig. 4 shows the modeling of the routing structure of the flip-chip package. Let P be the set of wire-bonding pads, and let B be the set of bump pads. For practical applications, the number of bump pads is larger than or equal to the number of wire-bonding pads, i.e., $|B| \geq |P|$, and each bump pad can be assigned to more than one wire-bonding pad. Let $R_b = \{r_1^b, r_2^b, \ldots, r_m^b\}$ be a set of m bump pad rings in the center of the package, and let $R_p = \{r_1^p, r_2^p, \ldots, r_k^p\}$ be a set of k wire-bonding pad rings at the boundary of the package. Each bump pad ring r_i^b consists of a set of q bump pads $\{b_1^i, b_2^i, \ldots, b_q^i\}$, and each wire-bonding pad ring r_j^p consists of l wire-bonding pads $\{p_1^j, p_2^j, \ldots, p_l^j\}$. Let N be the set of nets (could be two-pin or multipin nets) for routing. Each multipin net n in N is defined by a set of wire-bonding pads and a set of bump pads that should be connected. Each two-pin net can

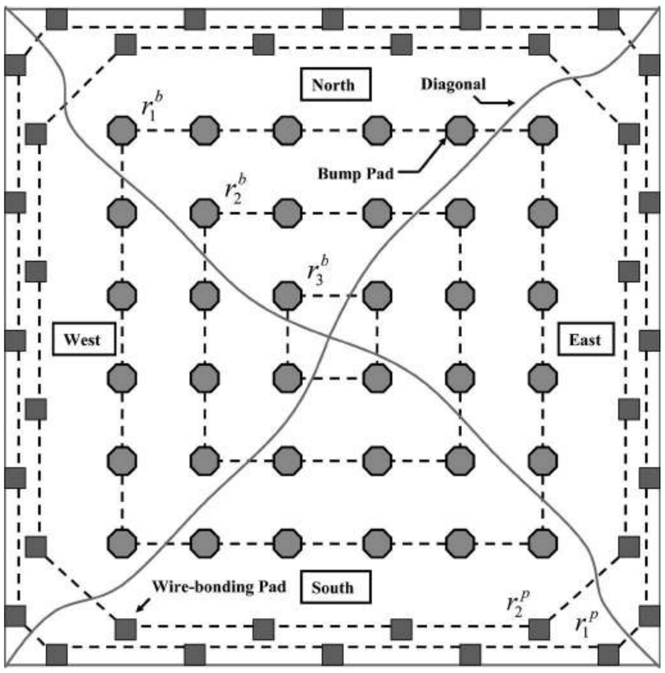


Fig. 4. Four sectors in a flip-chip package.

be assigned to a bump pad not included in the sets of bump pads for the multipin nets. Since the RDL routing for current technology is typically on a single layer, it does not allow *wire crossings*, for which two wires intersect each other in the routing layer. As shown in Fig. 4, based on the two diagonals of the flip-chip package, we partition the whole package into four sectors, namely: 1) $North = \{P_N, B_N, R_p^N, R_b^N\}$; 2) $East = \{P_E, B_E, R_p^E, R_b^E\}$; 3) $South = \{P_S, B_S, R_p^S, R_b^S\}$; and 4) $West = \{P_W, B_W, R_p^W, R_b^W\}$, where $P_i(B_i)$ and $R_p^i(R_b^i)$, $i \in \{N, E, S, W\}$ are the set of the wire-bonding (bump) pads and the set of the wire-bonding (bump) pad rings in the i sector, respectively. For practical applications, the wire-bonding pads in one sector only connect to the bump pads in the same sector.

We define an *interval* to be the segment between two adjacent bump pads in the same ring r_i^b or the segment between two adjacent wire-bonding pads in the same ring r_j^p. Given a flip-chip routing instance, there are two types of routing, namely: 1) the *monotonic routing* and 2) the *nonmonotonic routing*. A monotonic routing can be formally defined as follows.

Definition 1: A monotonic routing is a routing such that for each net n connecting from a wire-bonding pad p to a bump pad b, n intersects exactly one interval in each ring r_i^b and exactly one interval in each ring r_j^p.

As shown in Fig. 5(a), the nets n_2 and n_4 are monotonic routes. If we exchange the positions of two bump pads b_2 and b_4, the routings of n_2 and n_4 are nonmonotonic, as shown in Fig. 5(b). The wirelengths of the nets n_2 and net n_4 are increased. This shows a drawback of the nonmonotonic routing. Since the nonmonotonic routing occupies more routing resource, it causes significant problems for the single-layer routing. Thus, a good flip-chip package routing should be a monotonic routing without detours, as shown in Fig. 6, because the monotonic routing results in smaller total wirelength and higher routing completion, as compared to the nonmonotonic

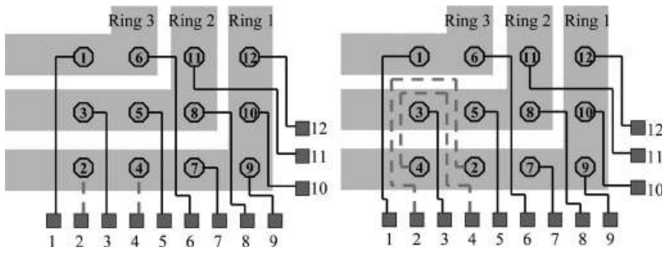


Fig. 5. (a) Monotonic routing. (b) Nonmonotonic routing.

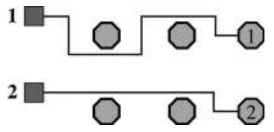


Fig. 6. Monotonic routing with and without detours.

routing. Furthermore, the signal skew, i.e., the difference of wirelength between the longest net and the shortest one, should also be considered for routing on the flip-chip package.

Based on the aforementioned definition, the routing problem can be formally defined as follows.

Problem 1: The single-layer flip-chip routing problem is to connect a set of $p \in P$ and a set of $b \in B$ so that no wire crosses each other, the routing is monotonic, and the total wirelength and the signal skew are minimized.

III. Routing Algorithm

In this section, we present our routing algorithm. First, we give the overview of our algorithm. Then, we detail the methods used in each phase.

A. Algorithm Overview

According to the routing flow shown in Fig. 7, our algorithm consists of two phases, namely: 1) global routing based on the *MCMF algorithm* [5] and 2) detailed routing based on the cross-point assignment, the net ordering determination, and the track assignment.

In the first phase, we construct four flow networks, namely: 1) G_N; 2) G_E; 3) G_S; and 4) G_W, one for each sector, to solve the assignment of the wire-bonding pads to the bump pads. Since we have only one layer for routing, the assignment should not create any wire crossings. We avoid the wire crossings by restricting the edges in the networks not to intersect each other. We first consider two-pin nets and then multipin nets. The reason is that multipin nets allow more than one wire-bonding pad to connect to one bump pad. Thus, the multipin nets may block the two-pin nets. Under this condition, a wire-bonding pad may not find a global path. Thus, the two-pin nets need to be considered first. We will detail the reason in Section III-B4. After applying MCMF, we obtain the flows representing the routes from wire-bonding pads to bump pads for the nets. Those flows give the global paths for the nets.

In the second phase, we use the cross-point assignment, the net ordering determination, and the track assignment to determine detailed routes. A *cross point* is the point for a net to pass through an interval. First, we find the cross points for

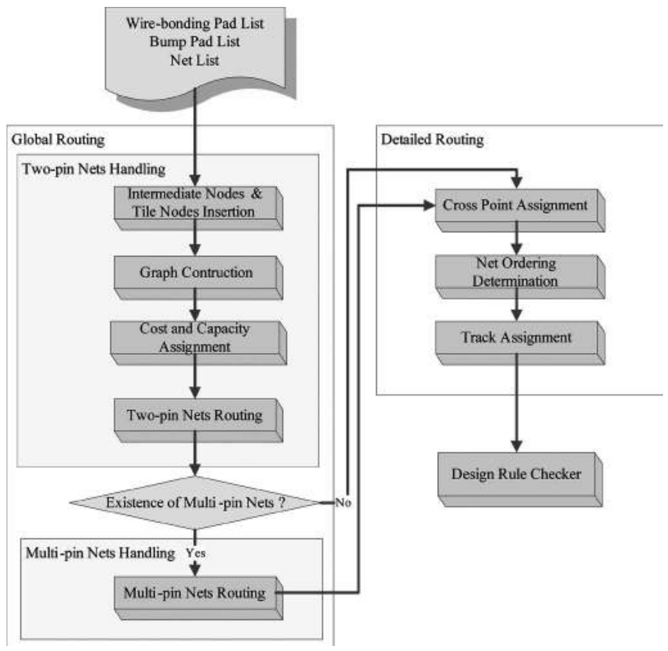


Fig. 7. RDL routing flow.

```
Algorithm: RDL Routing($P$, $B$, $N$)
$P$: set of all wire-bonding pads;
$B$: set of all bump pads;
$N$: set of all nets;
1 begin
2   Construct four graphs $G_N$, $G_E$, $G_S$, $G_W$ with only
3       2-pin nets;
4   Apply MCMF to find the assignment of each $p \in P$ to
5       $b \in B$ in the same sector and the global path
6       for each 2-pin net;
7   Add additional edges to represent the multi-pin net in the
8       four graphs;
9   Apply MCMF to find the assignment of each $p \in P$ to
10      $b \in B$ in the same sector and the global path
11      for each multi-pin net;
12  Find all cross points in all intervals for each net $n \in N$;
13  for the outermost ring $r_i^p$ to the innermost ring $r_j^b$
14      $S \leftarrow$ Net_Ordering_Determination();
15      // $S$ contains the routing sequence;
16      Track_Assignment($S$);
17 end
```

Fig. 8. Overview of the RDL routing algorithm.

all nets passing through the same interval. For all nets that pass through the same interval, we evenly distribute these cross points. Second, we use the net ordering determination technique presented in [7] to create the routing sequence between two adjacent rings so that we can guarantee to route all nets. Finally, we assign at least one track to each net based on the routing sequence obtained from the net ordering determination algorithm. Fig. 8 summarizes our routing algorithm.

B. Global Routing

In this section, we first show the basic flow network formulation. Then, we detail the capacity of each edge, the intermediate nodes, the tile nodes, and the cost of each edge. Finally, we discuss how to handle the multipin nets.

1) Basic Network Formulation: We describe how to construct the flow network G_S to perform the *concurrent assignment* for the *South* sector. The other three sectors can be processed similarly. As shown in Fig. 9(a), we define $D_S = \{d_1^S, d_2^S, \ldots, d_h^S\}$ to be a set of h *intermediate nodes*. Each intermediate node represents an interval $(p_y^j, p_{y+1}^j)((b_x^i, b_{x+1}^i))$ in a wire-bonding (bump) pad ring. $T_S = \{t_1^S, t_2^S, \ldots, t_u^S\}$ is a set of u *tile nodes*. Each tile node represents a tile $(p_y^j, p_{y+1}^j, p_{y'}^{j+1}, p_{y'+1}^{j+1})((b_x^i, b_{x+1}^i, b_{x'}^{i+1}, b_{x'+1}^{i+1}))$ between two adjacent wire-bonding (bump) pad rings. We construct a graph $G_S = (P_S \cup D_S \cup B_S \cup T_S, E)$ and add a source node s and a sink node t to G_S. Each intermediate node d has a capacity of K_d, where K_d represents the maximum number of nets that are allowed to pass through an interval d. Each tile node t has a capacity of L_t, where L_t represents the maximum number of nets that are allowed to pass through a tile t. We will detail how to handle the capacity of the intermediate nodes and the tile nodes so that MCMF can be applied in Section III-B2. There are 11 types of edges:

1) edges from a wire-bonding pad to a bump pad;
2) edges from a wire-bonding pad to an intermediate node;
3) edges from a wire-bonding pad to a tile node;
4) edges from an intermediate node to a bump pad;
5) edges from an intermediate node to another intermediate node;
6) edges from an intermediate node to a tile node;
7) edges from a tile node to a bump pad;
8) edges from a tile node to an intermediate node;
9) edges from a tile node to another tile node;
10) edges from the source node to a wire-bonding pad;
11) edges from a bump pad to the sink node.

Each edge is associated with a $(cost, capacity)$ tuple to be described in the following sections. Recall that we do not allow wire crossings for all wires. Since E represents the possible global paths for all nets, we can guarantee that no wire crossings will occur if there are no crossings in edges. Thus, we construct all the edges and avoid crossings of all edges at the same time. Fig. 9(b) shows an example flow network G_S for the *South* sector. The last two types of edges are not shown here. Furthermore, we do not construct edges between the two tile nodes in the center of the two wire-bonding pad rings because the placement of these tile nodes is symmetric. We can solve MCMF in time $O(|V|^2\sqrt{|E|})$ based on the network flow algorithm presented in [5], where V is the vertex set in the flow network.

Theorem 1: Given a flow network with the vertex set V and edge set E, the global routing problem can be solved in $O(|V|^2\sqrt{|E|})$ time.

Proof: Immediate from the aforementioned discussions. ■

2) Capacity Assignment and Node Construction: Now, we introduce the capacity of each edge, the intermediate nodes, and the tile nodes. Fig. 10 shows the capacity and cost for all 11 types of edges in the complete flow network. For an edge e, if e is from a wire-bonding pad to a bump pad, an intermediate node, or a tile node, the capacity of e is set to one. If e is from an intermediate node or a tile node to a bump pad b, then the capacity of e is set to M_b, where M_b is the maximum number

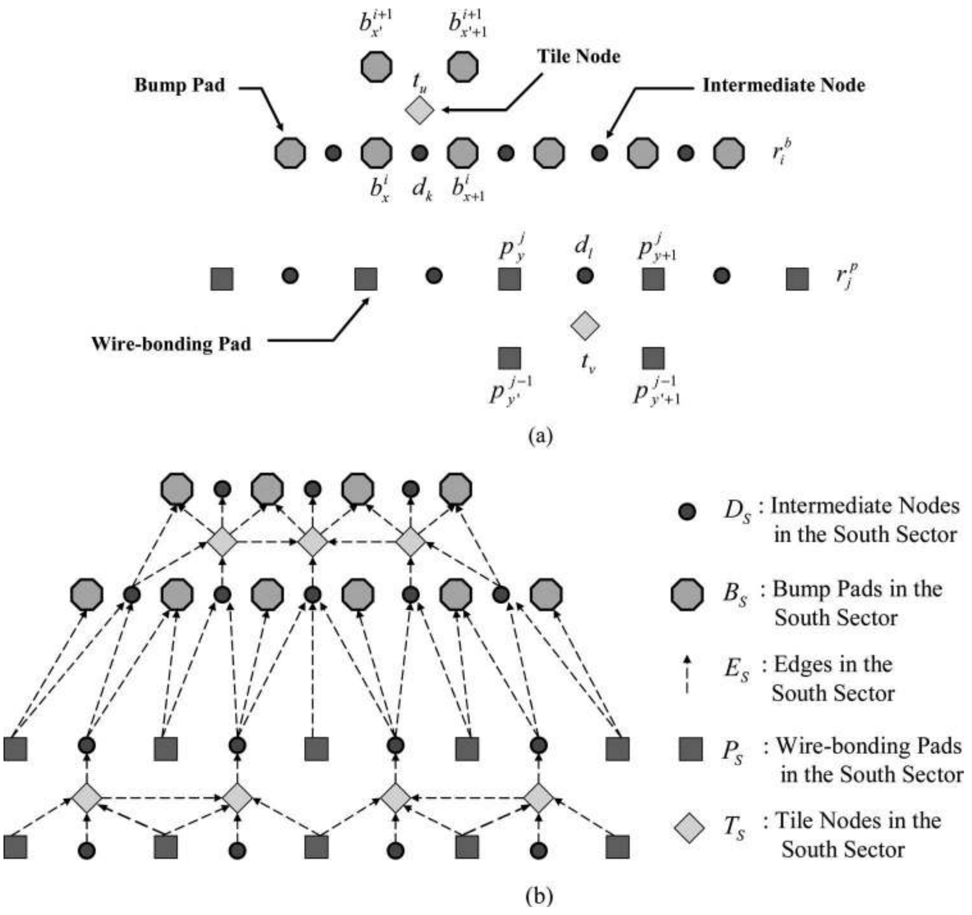

Bump Pad
Tile Node
Intermediate Node
Wire-bonding Pad
(a)
(b)
D_S : Intermediate Nodes in the South Sector
B_S : Bump Pads in the South Sector
E_S : Edges in the South Sector
P_S : Wire-bonding Pads in the South Sector
T_S : Tile Nodes in the South Sector

Fig. 9. (a) Intermediate nodes and tile nodes. (b) Flow network for the *South* sector.

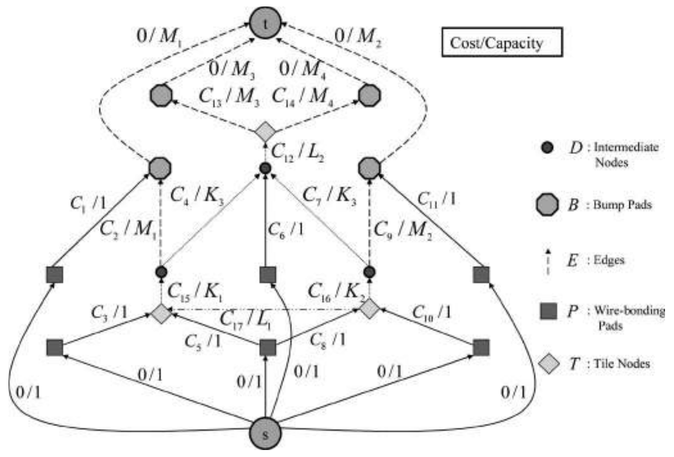


Fig. 10. Capacity and cost on edges.

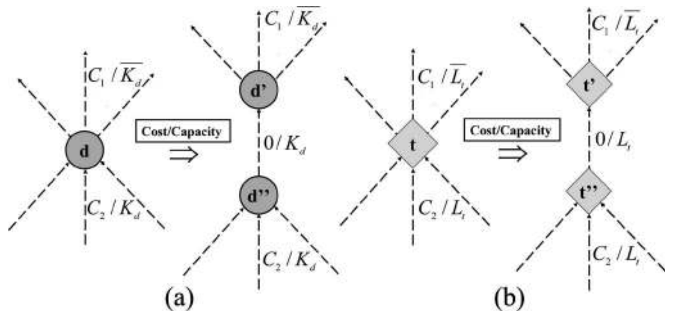


Fig. 11. (a) Capacity and cost on intermediate nodes. (b) Capacity and cost on tile nodes.

of nets that are allowed to connect to the bump pad b. Recall that an intermediate node d has a capacity of K_d, where K_d is the maximum number of nets that are allowed to pass through this intermediate node d. This means that the capacity of each incoming edge of an intermediate node d is equal to K_d. If e is an incoming edge of a tile node t, then the capacity of e is set to L_t, where L_t is the maximum number of nets that are allowed to pass through the tile node t. As shown in Fig. 11, in order to model this situation, we decompose each intermediate node d into two intermediate nodes d' and d'', and an edge is connected from d'' to d', with a capacity of K_d. All outgoing edges of d are now connected from d', with a capacity of $\bar{K}_d$, and all incoming edges of d are now connected to d'', with a capacity of K_d. Each tile node t is also decomposed into two tile nodes t' and t'', and the capacity of a tile node t is set to L_t, where L_t is the maximum number of nets that are allowed to pass through this tile node t. The capacity of the edges from the source node to the wire-bonding pads is set to one, and the capacity of the edges from each bump pad b to the sink node is set to M_b. There are three worst cases of congestion in a tile, as shown in Fig. 12. The four nodes in the three figures are all bump pads. In Fig. 12(a) and (c), the maximum number of nets passing through the tile is $2K$. In Fig. 12(b), the maximum number of nets passing through the tile is $3K$. If we do not use the tile node t, the maximum number of nets in Fig. 12(a)–(c)

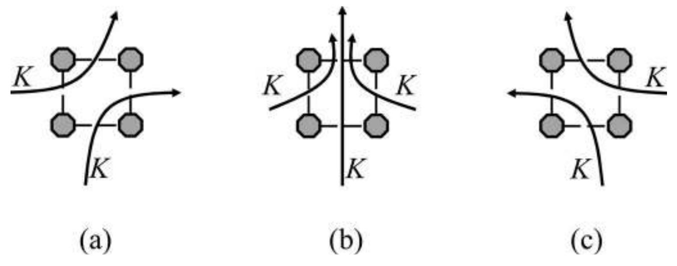


Fig. 12. Three kinds of congestion in a tile.

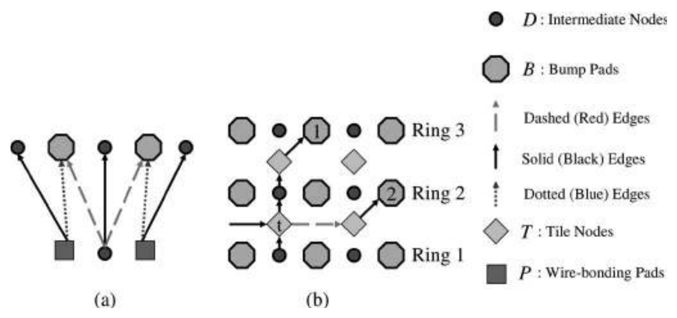


Fig. 13. Adjustment of α values on edges in the *South* sector.

could exceed the capacity of a tile ($2K > L_t$ or $3K > L_t$). Since the capacity of each tile node is well modeled in our flow network, we can totally avoid this congestion problem.

3) Cost of Edges: The cost function of each edge is defined by the following equation:

$$\text{Cost} = \alpha \times W_L \tag{1}$$

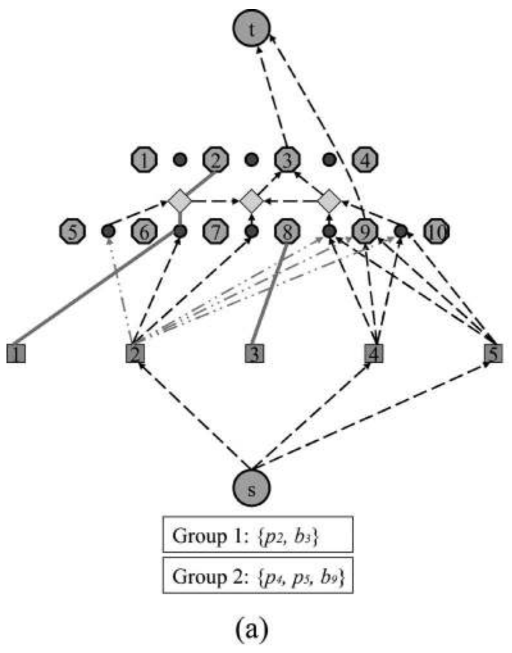
where W_L denotes the Manhattan distance between two terminals of an edge, and α is an adaptive parameter to adjust the cost of different types of edges. By adjusting the value of α, we can control the wirelength of each net to avoid large signal skews among different nets. As an example shown in Fig. 13(a), we assign the smallest α to the dashed (red) edge that connects an intermediate node to a bump pad to assign the intermediate node to the bump pad first. By doing so, the routing for a net starting from a preceding ring can be completed earlier to reduce its routing length (and, thus, signal skew). As an example shown in Fig. 13(b), the dashed (red) edge that connects one tile node to another tile node is also assigned the smallest α to assure that fewer bump pad rings are used. Since the wirelength between the tile node t and the bump pad 1 is the same as that between t and the bump pad 2, we have to assign the smallest α to the dashed (red) edge to make t connect the bump pad 2 first. Thus, we can reduce the number of long nets to reduce the signal skew by using fewer bump pad rings. If a wire-bonding pad is assigned to a bump pad directly, it might generate a very short net. Hence, we assign the largest α to the dotted (blue) edge that connects a wire-bonding pad to a bump pad to avoid too short connections between the two types of pads to reduce the signal skew. Finally, the solid (black) edge that connects two intermediate nodes, a tile node to a bump pad, or an intermediate node to a tile node is assigned a medium α. Since the solid (black) edge does not influence the signal skew, the medium α is set to one. The costs of the edges from the source node to the wire-bonding pads and the costs of the edges from the bump pads to the sink node are both set to zero.

4) Multipin Net Handling: For practical flip-chip routing, if a bump pad can be assigned to any two or even more wire-bonding pads, we can just increase the capacity of the bump pad to connect more wire-bonding pads. Since we construct the edges for the two-pin nets and the multipin nets simultaneously, the global routing result is optimal. However, for other practical flip-chip routing, a net may connect multiple wire-bonding pads (which are assigned the same signal such as power or ground pads) to a bump pad. This bump pad cannot be assigned to other nets. As stated before, we first assign two-pin nets and then multipin nets. We only construct the edges associated with the two-pin nets and apply MCMF for the assignment. After the assignment, we delete all edges from the source node s and all edges to the sink node t. (However, the flows of the edge e, the intermediate node d, and the tile node t for each assigned two-pin net will be kept in the flow network.) The global paths of the assigned two-pin nets are not deleted and considered as blockages F during the construction of the edges for the multipin nets. Recall that if there are no edge crossings in the flow network, then there are no wire crossings in the final routing solution. When we construct the edges for the multipin nets, an edge e exists only if e does not intersect any blockages or never crosses the assigned two-pin nets. Then, we add the edges from the source node to the wire-bonding pads associated with the multipin nets and the edges from the bump pads associated with the multipin nets to the sink node. Fig. 14(a) illustrates an example. We assume that a multipin net n consists of $((p_2, p_4, p_5), (b_3, b_9))$, which means that three wire-bonding pads 2, 4, and 5 are only free to be assigned to one of the two bump pads 3 and 9. No other wire-bonding pads can be assigned to these two bump pads. Redundant edges are deleted by the blockage f_i. For example, the edge from p_2 to the intermediate node between b_8 and b_9 is deleted because it intersects the blockage (p_3, b_8). By using MCMF, the wire-bonding pads and bump pads are grouped into two sets: $\{p_2, b_3\}$ and $\{p_4, p_5, b_9\}$. Fig. 14(b) illustrates why we handle two-pin nets first. In this example, we assume that only the bump pad 1 for two-pin nets and the bump pad 2 for multipin nets can be assigned to wire-bonding pads. If we handle the multipin net 2 first, then the two-pin net 1 cannot be assigned to the bump pad 1 to find a global path. The reason is that the multipin net 2 divides the region into two subregions and blocks the wire-bonding pad 1. In order to avoid this situation, we shall handle two-pin nets first. The similar idea is applied in the planar routing. As shown in Fig. 15(a), in a planar routing, if a net such as net 1 or net 2 is routed to divide the region into two subregions, it should be routed later. Otherwise, as shown in Fig. 15(b), other nets such as net 3 or net 4 may cross the net.

Based on the global routing algorithm, we have the following theorem.

Theorem 2: Given a set of wire-bonding pads, a set of bump pads, and a set of nets, if there exists a feasible solution computed by the MCMF algorithm, we can guarantee 100% detailed routing completion.

Proof: In our global routing model, MCMF is optimal for two-pin nets and suboptimal for multipin nets. Since we



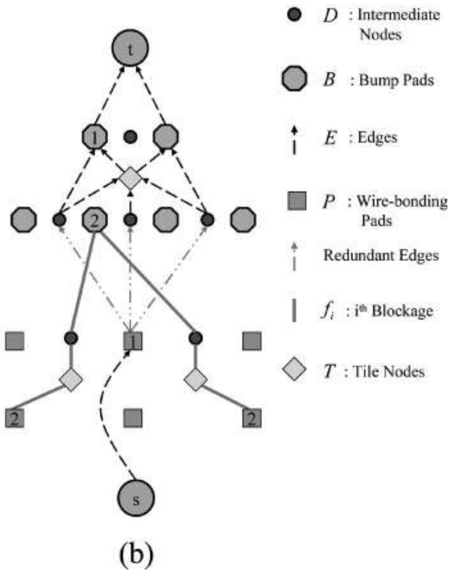


Fig. 14. (a) Assign multipin nets. (b) Handle multipin nets first.

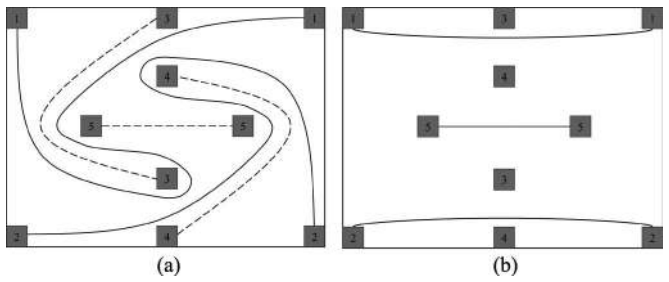


Fig. 15. (a) Routing sequence: $\{5, 3, 4, 1, 2\}$. (b) Routing sequence: $\{1, 2, 3, 4, 5\}$.

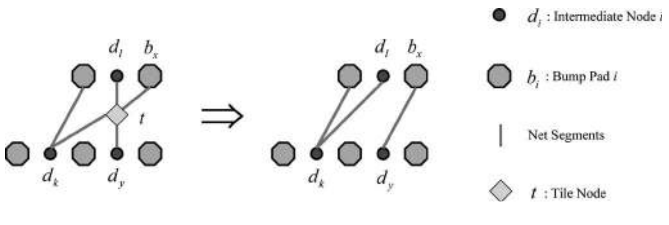


Fig. 16. Redefined global paths.

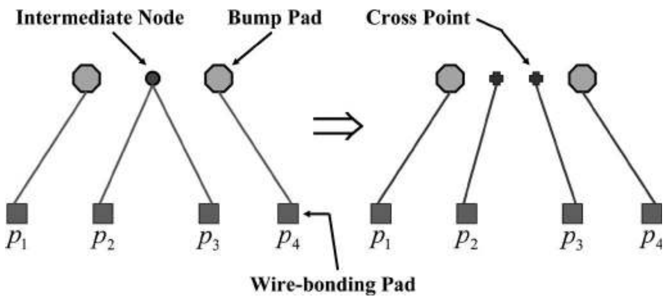


Fig. 17. Cross-point assignment.

consider the routing resource in the global routing stage and will never assign nets to exceed the capacity of an interval or a tile, we will never violate the design rules. Also, because we do not allow edge crossings during the flow network construction, the final routing solution will not generate wire crossings. Thus, after the assignment, all global paths are routable in the detailed routing stage. ■

C. Detailed Routing

In this section, we explain the three methods used in our detailed routing. As shown in Fig. 16, after the global routing, each global path contains only wire-bonding pads, intermediate nodes, and bump pads. The two global paths $\langle d_k, t, d_l\rangle$ and $\langle d_y, t, b_x\rangle$, which pass through the tile node t, are remodeled as $\langle d_k, d_l\rangle$ and $\langle d_y, b_x\rangle$. Tile nodes are not needed for the final representations of the global paths because a tile node is just used to avoid the congestion overflow.

1) Cross-Point Assignment: Based on the global routing result (discussed in Section III-B), we use the cross-point assignment algorithm to evenly distribute nets that pass through the same interval (see Fig. 17 for an example). As shown in Fig. 17, the two nets from wire-bonding pads p_2 and p_3 pass through the same intermediate node. Thus, we split the intermediate node into two cross points. Since the maximum number of intermediate nodes is $((|B| - |R_b|) + (|P| - |R_p|))$, we have the following theorem.

Theorem 3: The cross-point assignment problem can be solved in $O(|B| + |P|)$ time.

Proof: If there are q_i bump pads in the bump pad ring r_i^b in the *South* sector, there will be $(q_i - 1)$ intervals of the bump pad ring r_i^b. Hence, there will be $(q_i - 1)$ intermediate nodes. Since the number of bump pad rings is $|R_b^S|$, the maximum number of intermediate nodes is $\sum_{i=0}^{|R_b^S|}(q_i - 1) = |B_S| - |R_b^S|$. As for the wire-bonding pads, the condition is the same as that of the bump pads, and the conditions of the remaining three sectors are the same as those of the *South* sector. Thus, the maximum number of intermediate nodes is $((|B| - |R_b|) + (|P| - |R_p|))$, and the time complexity is $O(|B| + |P|)$ (the upper bound of

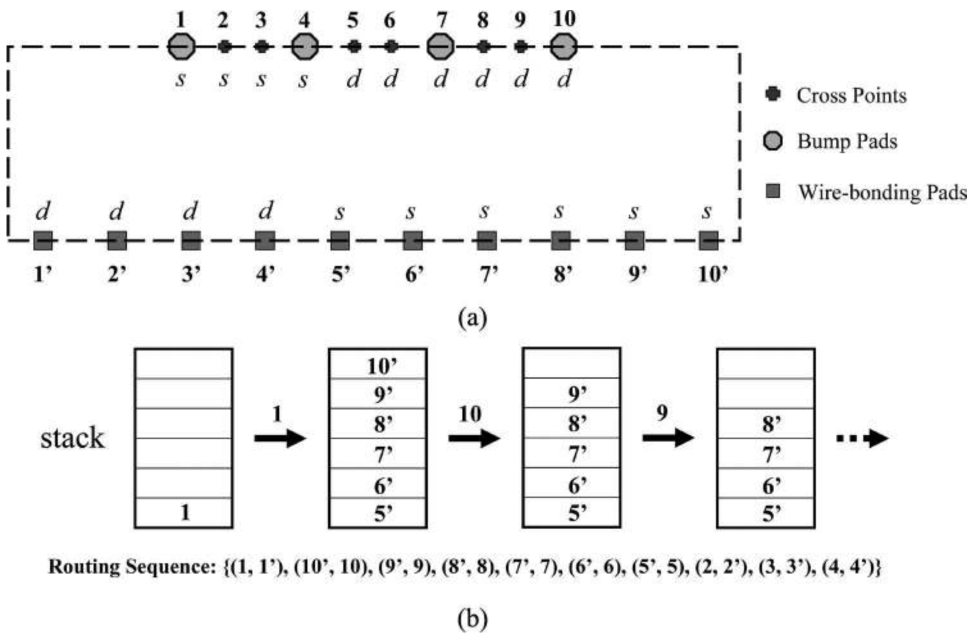


Fig. 18. (a) Net segments between two adjacent rings. (b) Stack for net ordering determination.

$((|B| - |R_b|) + (|P| - |R_p|)))$ to assign cross points to each intermediate node. ■

2) Net Ordering Determination: After the assignment of cross points, each net has its path to cross each interval. For two adjacent rings, we can treat the routing between the two rings as a channel routing. Thus, we can use the net ordering determination algorithm presented in [7] to generate a routing sequence $S = \langle (n_1^s, n_1^d), (n_2^s, n_2^d), \ldots, (n_k^s, n_k^d) \rangle$, with k net segments. Each net segment $n_i(j, j')$ is represented by a source–destination pair/tuple (n_i^s, n_i^d). We first determine the source and destination for each net based on the counterclockwise traversing distance along the leftmost and rightmost boundaries. If the counterclockwise traversing distance along the leftmost boundary is shorter than the counterclockwise traversing distance along the rightmost boundary, the terminal j is a source, and the terminal j' is a destination. Otherwise, the terminal j is a destination, and the terminal j' is a source. For example, given the net 1 shown in Fig. 18(a), since the counterclockwise traversing distance along the leftmost boundary is shorter than the counterclockwise traversing distance along the rightmost boundary, we make the terminal 1 a source and the terminal $1'$ a destination. For the net 10, however, since the counterclockwise traversing distance along the leftmost boundary is longer than the counterclockwise traversing distance along the rightmost boundary, we make the terminal $10'$ a source and the terminal 10 a destination. Starting from an arbitrary terminal, we then generate a circular list for all terminals ordered counterclockwise according to their positions on the boundaries. A stack is used to check if there exist crossovers among the net segments. For each terminal of net segment n_i, if it is a source, then we push it into the stack. If this terminal is a destination and the top element of the stack belongs to the same net segment, then net segment n_i is matched, and the top element is popped. Otherwise, if the stack is empty, or this terminal is a destination and the top element of the stack does not belong to the same net segment, then we search the circular list for the next terminal. We keep searching the circular list until all nets are matched. As shown in Fig. 18(b), we start with the terminal 1. Since the terminal 1 is a source, we push it into the stack. Then, we search each terminal on the boundary counterclockwise. The terminal $1'$ is searched, and it is a destination; thus, we compare it with the top element of the stack. Because these two terminals belong to the same net segment, we pop the top element and determine the routing sequence of the net segment n_1. Keeping on searching, since the terminals $2'$, $3'$, and $4'$ are all destinations, we do not push them into the stack. Since the terminals $5'$, $6'$, $7'$, $8'$, $9'$, and $10'$ are all sources, we push them into the stack. Then, we process the terminal 10, which is a destination and matches the top element in the stack. Thus, we pop the net segment n_{10} and add it into the routing sequence. Repeating this step, we can get the resulting routing sequence. With this sequence S, we can guarantee that each net segment between two adjacent rings can be routed without intersecting each other. For example, given an instance shown in Fig. 18(a), according to the above-described net ordering determination algorithm, we can obtain the sequence $S = \langle (n_1, n_1'), (n_{10}', n_{10}), (n_9', n_9), (n_8', n_8), (n_7', n_7), (n_6', n_6), (n_5', n_5), (n_2, n_2'), (n_3, n_3'), (n_4, n_4') \rangle$. According to the net ordering determination algorithm, we have the following theorem.

Theorem 4: Given a set N of nets, the net ordering determination problem can be solved in $O(|N|^2)$ time.

Proof: According to the net ordering determination algorithm, the worst case happens when only one net is matched during each searching cycle. In this case, the total number of terminal searches is $\sum_{i=0}^{|N|-1} 2(|N| - i) = |N|^2 + |N|$. Hence, the time complexity is $O(|N|^2)$. ■

3) Track Assignment: With the net ordering, we can use maze routing to route all nets for any two adjacent rings. However, maze routing is quite slow and generates too many bends. (For example, for a small circuit with 513 nets, we need 25 min on a 1.2-GHz SUN Blade 2000 workstation with 8-GB

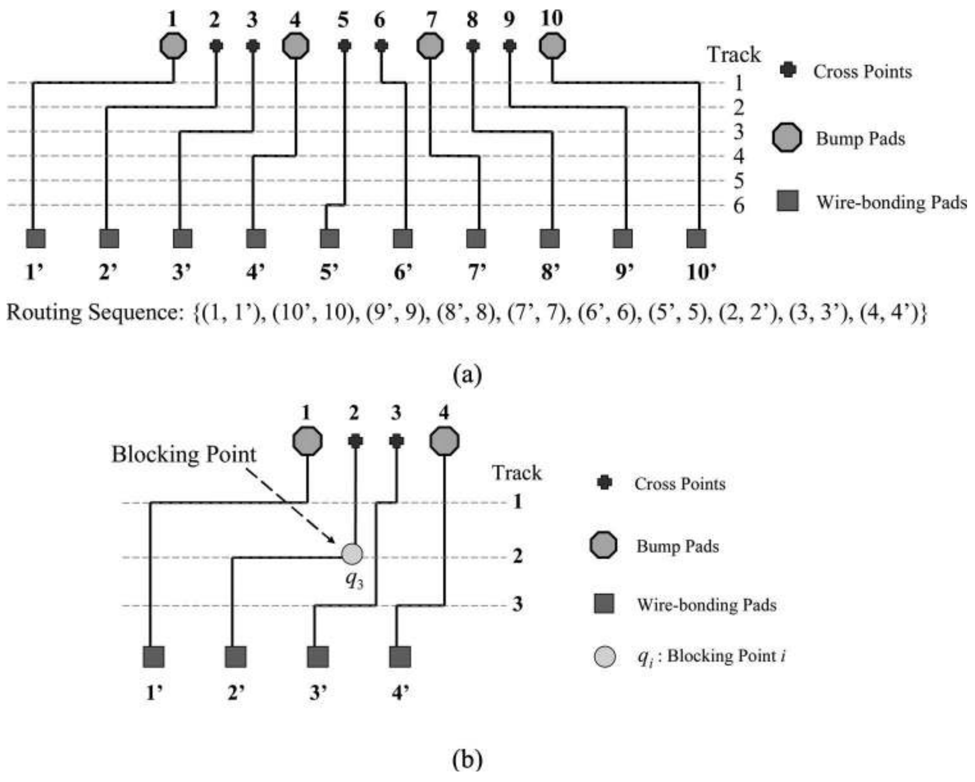

1 2 3 4 5 6 7 8 9 10
Track
1 2 3 4 5 6
Cross Points
Bump Pads
Wire-bonding Pads
1' 2' 3' 4' 5' 6' 7' 8' 9' 10'
Routing Sequence: {(1, 1'), (10', 10), (9', 9), (8', 8), (7', 7), (6', 6), (5', 5), (2, 2'), (3, 3'), (4, 4')}
(a)
Blocking Point
q_3
q_i : Blocking Point i
(b)

Fig. 19. (a) Example for track assignment. (b) Blocking point.

memory to complete the detailed routing.) Thus, we propose a track assignment algorithm to assign tracks to each net segment of any two adjacent rings. For each net segment n_i in S, according to the relative locations of n_i^s and n_i^d, we search a track to be assigned to n_i from the top to the bottom or from the bottom to the top. We search the tracks from the top to the bottom if n_i^s is on the top-right side of n_i^d or n_i^s is on the bottom-right side of n_i^d. Otherwise, we search the tracks from the bottom to the top. If we find a track h and it does not create any overlap with other wires, then we assign h to n_i. As shown in Fig. 19(a), we assign net segment n_1 first. Since the terminal 1 is a source and the net ordering determination algorithm makes each net routed counterclockwise from the source to the destination along the boundary, we search from track 1 to track 6. Thus, n_1 is assigned to track 1 first. Since the terminal $5'$ is a source, we search from track 6 to track 1. Thus, n_5 is assigned to track 6 first. Also, we record the blocking points Q for n_i. A *blocking segment* is a wire on track $h+1$ (if we search from the top to the bottom) or $h-1$ (if we search from the bottom to the top) to stop n_i from being assigned to $h+1$ or $h-1$ without creating any overlap with it. A *blocking point* q_i is a terminal of the blocking segment whose projection on h overlaps with n_i. As shown in Fig. 19(b), the point q_3 on track h_2 is the blocking point for net n_3. If we cannot find such h, we rip up and reroute all net segments n_1 to n_{i-1}. For each net segment n_k to be rerouted, we use the concept of the dogleg in the channel routing to break a segment into two segments based on the blocking point q_k, such as q_3 in Fig. 19(b). Then, we assign the segment that will not overlap with q_k on the lowest possible track (if we search from the top to the botto m) or on the highest possible track (if we search from the bottom to the top). After assigning tracks, we record the new blocking points for n_k. Note that since, now, each net segment may be assigned with more than one track, we may have more than one blocking point for each net. Fig. 20 summarizes the track assignment algorithm. According to this algorithm, we have the following theorem.

```
Algorithm: Track Assignment($S_j$, $H$)
$S_j$: a routing sequence between rings $r_j$ and $r_{j+1}$;
$H$: the maximum number of tracks;
1 begin
2   for each net segment $n_i$ in $S_j$
3       Let $(x_i^s, y_i^s)$ $((x_i^d, y_i^d))$ be the coordinate of the
4           source (destination) of $n_i$;
5       if $((x_i^s \geq x_i^d$ and $y_i^s \geq y_i^d)$ or $(x_i^s \geq x_i^d$ and $y_i^s \leq y_i^d))$
6           Find a track $h$ of $H$ from the top to the bottom
7               without creating an overlap with other wires;
8       else
9           Find a track $h$ of $H$ from the bottom to the top
10              without creating an overlap with other wires;
11      if such $h$ exists
12          Assign $h$ to $n_i$;
13      else
14          for all pre-routed net segment $n_k$
15              Divide into two segments according
16                  to the blocking point $q_k$;
17              Assign the segment not overlapping with $q_k$
18                  to the first available track along the cur-
19                  rent search direction (from top to bottom
20                  or bottom to top);
21 end
```

Fig. 20. Algorithm for track assignment.

Theorem 5: Given a set N of nets and the number of tracks H, the track assignment problem can be solved in $O(|N|^2H(|R_b| + |R_p|))$ time.

Proof: The worst case of the track assignment algorithm is that we have to rip up and reroute every time while assigning the next net. Thus, the total number of times to assign nets to tracks is $\sum_{i=1}^{|N|} i = |N|^2 + |N|/2$. When assigning a net, the maximum number of track searches is H, and the number of channels is $(|R_p| + |R_b| - 1)$. Hence, the time complexity is $O(|N|^2H(|R_b| + |R_p|))$. ■

D. Complexity Analysis

If there are $|B|$ bump pads, $|P|$ wire-bonding pads, $|D|$ intermediate nodes, $|T|$ tile nodes, $|R_b|$ bump pad rings, and $|R_p|$ wire-bonding pad rings, we can construct a flow network composed of $|V|$ vertices and $|E|$ edges for the global routing, where $|V| = |B| + |P| + |D| + |T|$ and $|E| =$ edges among these vertices. In the detailed routing, there are $|N|$ global paths, and each channel is divided into H tracks. Hence, the time complexity is as follows: $O(|V|^2\sqrt{|E|})$ (global routing)$+$ $O(|B|+|P|)$ (cross-point assignment)$+ O(|N|^2)$(net ordering determination) $+ O(|N|^2H(|R_b| + |R_p|))$ (track assignment) $= O(|V|^2(\sqrt{|E|} + H(|R_b| + |R_p|)))$. The space complexity is $O(|E|)$ since we have $O(|B| + |P| + |D| + |T|)$ nodes and $O(|E|)$ edges. Thus, we can solve the RDL routing problem in polynomial time.

Theorem 6: Given a set P of wire-bonding pads, a set B of bump pads, and a set N of nets, if there exists a feasible solution computed by the RDL routing algorithm, the RDL routing problem can be solved in $O(|V|^2(\sqrt{|E|} + H(|R_b| + |R_p|)))$ time and $O(|E|)$ space.

Proof: The time complexity analysis is immediate from Theorems 1, 3, 4, and 5. First, since $|V| = |B|+|P|+|D|+|T|$, the time complexity of the global routing $O(|V|^2\sqrt{|E|})$ dominates that of the cross-point assignment $O(|B| + |P|)$. Second, the time complexity of the track assignment $O(|N|^2H(|R_b| + |R_p|))$ dominates that of the net ordering determination $O(|N|^2)$. Finally, since $|N| = |P|$, the time complexity $O(|V|^2)$ dominates $O(|N|^2)$. Thus, the time complexity of the RDL routing algorithm is given by $O(|V|^2(\sqrt{|E|} + H(|R_b| + |R_p|)))$. The space complexity of pads $O(|B| + |P|)$ dominates that of nodes $O(|D| + |T|)$. Since there is an edge from the source node s to every wire-bonding pad of P, and there is an edge from every bump pad of B to the sink node t, the space complexity of edges $O(|E|)$ dominates that of pads $O(|B| + |P|)$. Thus, we can reduce the space complexity from $O(|B| + |P| + |D| + |T| + |E|)$ to $O(|E|)$. ■

IV. Experimental Results

We implemented our algorithm in the C++ programming language on a 1.2-GHz SUN Blade 2000 workstation with 8-GB memory. The benchmark circuits, which are listed in Table I, are real industry designs. In Table I, "Circuits" denotes the names of circuits; "#Nets" denotes the number of nets; "$\#R_p$" denotes the number of wire-bonding pad rings; "$\#p$" denotes the number of wire-bonding pads; "$\#R_b$" denotes the number of bump pad rings; and "$\#b$" denotes the number of bump pads. In each of fs900, fs2116, and fs4096, the number of wire-bonding pads equals the number of bump pads. Thus, each wire-bonding pad needs to be assigned to exactly one bump pad. Hence, these three cases are more difficult for routing than the other four cases.

In Table II, we show how to calculate the values of M_b, K_d, H_i, and L_t. As defined in the previous sections, M_b is the maximum number of nets allowed to connect to a bump pad b, K_d is the maximum number of nets allowed to pass through an intermediate node d, H_i is the maximum number of tracks between two adjacent pad rings i and $i + 1$, and L_t is the maximum number of nets allowed to pass through a tile node t. All these variables can be expressed by the equations shown in the table. We calculate the values of these variables during the RDL routing process. The parameters used to calculate the M_b, K_d, H_i, and L_t variables are listed in Table III. They are all structure and design-rule related parameters.

Since there are no flip-chip routing algorithms in the literature, we compared our algorithm with the following heuristic currently used in industry. This heuristic is called the nearest node connection (NNC) algorithm. In NNC, the wires are routed sequentially. If a wire-bonding pad p can find a free bump pad b in a restricted area of the nearest bump pad ring r_m^b, then it connects p to b. If there are no free bump pads in r_m^b, then we search for a free bump pad in the next bump pad ring r_{m+1}^b. This process is repeated until we find a free bump pad.

The experimental results are shown in Table IV. We report the total wirelength, the critical wirelength (the wirelength of the longest net), the maximum signal skews, and the central processing unit times. Since the routability is guaranteed to be 100%, we do not report it. As compared with NNC, the experimental results show that our network-flow-based algorithm reduces the total wirelength by 10.2%, the critical wirelength by 13.4%, and the signal skews by 13.9%, in reasonably longer running time. Note that for fs2116 and fs4096, NNC fails to find a routing solution. In Fig. 21, the running time of our algorithm is plotted as a function of the number of nets. Empirically, the running time of our RDL routing algorithm approaches quadratic (about $N^{2.17}$) to the number of nets N, with the least square analysis for the log–log plot of the function. In Table V, we report the memory usage (in kilobytes) for each circuit for the RDL routing. In Fig. 22, the memory requirement of our algorithm is plotted as a function of the number of nets. The empirical memory complexity of our RDL routing algorithm is between linear and quadratic (about $N^{1.47}$) to the number of nets N, again with the least square analysis for the log–log plot of the function. The experimental results show that our network-flow-based RDL algorithm is effective and efficient for flip-chip designs. Fig. 23 shows the RDL routing result of fs900.

We also explore the effects of different α on wirelength and skew. In Table II, we give the equations for the computation of the upper bound of the smallest α and the lower bound of the largest α. The two bounds come from the geometric relation of the pad placement. Nets that are composed of the wire-bonding pads of the inner wire-bonding pad ring and the bump pads of the outer bump pad ring are often short. Thus, by modeling d_{PB} and d_{BB} into the equation of the lower bound of the largest α,

TABLE I
BENCHMARK CIRCUITS FOR RDL ROUTING

Circuits	#Nets (2-pin/multi-pin)	#Rp	#p	#Rb	#b
fs90b740	646/0	2	646	7	812
fsa0ac013aa	657/4	2	657	17	1156
fsa0ac015aa	639/6	2	639	17	1156
fwaa281	513/24	2	513	13	676
fs900	900/0	4	900	15	900
fs2116	2116/0	6	2116	23	2116
fs4096	4096/0	8	4096	32	4096

TABLE II
EXPERIMENTAL VARIABLES

Variable	Mathematical Expression
M_b	$M_b = \begin{cases} 1, & \text{for the bump pads that are connected by 2-pin nets} \\ j,\ j \geq 1, & \text{for the bump pads that are connected by multi-pin nets} \end{cases}$
K_d	$K_d = \begin{cases} \left\lfloor \frac{w_{IB} - 2\times s_{BW} - w_W}{w_W + s_{WW}} \right\rfloor + 1, & \text{for the intermediate nodes between two adjacency bump pads} \\ \left\lfloor \frac{w_{IP} - 2\times s_{PW} - w_W}{w_W + s_{WW}} \right\rfloor + 1, & \text{for the intermediate nodes between two adjacency wire-bonding pads} \end{cases}$
H_i	$H_i = \begin{cases} \left\lfloor \frac{d_{BB} - 2\times s_{BW} - w_W}{w_W + s_{WW}} \right\rfloor + 1, & \text{for the tracks between two adjacent bump pad rings} \\ \left\lfloor \frac{d_{PP} - 2\times s_{PW} - w_W}{w_W + s_{WW}} \right\rfloor + 1, & \text{for the tracks between two adjacency wire-bonding pads} \\ \left\lfloor \frac{d_{PB} - s_{PW} - s_{BW} - w_W}{w_W + s_{WW}} \right\rfloor + 1, & \text{for the tracks between a bump pad ring and a wire-bonding pad ring} \end{cases}$
L_t	$L_t = \min(H_i, K_d)$
Upper Bound of the Smallest α	$\textit{Upper Bound of the Smallest } \alpha = \frac{d_{PB}}{\lvert R_p \rvert \times \sqrt{d_{PB}^2 + \frac{1}{4} d_{PP}^2}}$
Lower Bound of the Largest α	$\textit{Lower Bound of the Largest } \alpha = \frac{\frac{1}{2} d_{PB} + \sqrt{\frac{1}{4} d_{PB}^2 + \frac{1}{4} d_{BB}^2} + \sqrt{d_{BB} + d_{PB}^2}}{d_{PB}}$

TABLE III
STRUCTURE AND DESIGN-RULE RELATED PARAMETERS

Parameter	Meaning
w_{IB}	the interval width between two adjacent bump pads
w_{IP}	the interval width between two adjacent wire-bonding pads
s_{BW}	the minimum spacing required between a bump pad and a wire
s_{PW}	the minimum spacing required between a wire-bonding pad and a wire
s_{WW}	the minimum spacing required between two wires
w_W	the wire width
d_{PP}	the distance between two adjacent wire-bonding pad rings
d_{BB}	the distance between two adjacent bump pad rings
d_{PB}	the distance between a wire-bonding pad ring and a bump pad ring

we can avoid short nets. Furthermore, we want to make the nets of the outer wire-bonding pad rings be assigned to the outer bump pad rings; thus, we model d_{PP} and d_{PB} into the equation of the upper bound of the smallest α. In order to minimize the signal skew, the largest α has to be larger than the lower bound, and the smallest α has to be smaller than the upper bound. Furthermore, we also observe that when the largest (smallest) α is scaled up (down), the critical wirelength

TABLE IV
RDL ROUTING RESULTS (N/A: NOT AVAILABLE)

Algorithm / Circuits	Total wirelength (μ m)			Critical wirelength (μ m)			Skew			CPU time (s)	
	NNC	Our method	Improve ment	NNC	Our method	Improve ment	NNC	Our method	Improve ment	NNC	Our method
fs90b740	814927	779089	4.6%	3682	3357	8.9%	3392	3067	9.6%	0.28	0.68
fsa0ac013aa	773717	700831	10.4%	5274	4539	13.9%	5139	4404	14.3%	0.39	0.87
fsa0ac015aa	699986	618363	13.2%	5254	4068	22.5%	5118	3932	23.2%	0.34	0.79
fwaa281	663762	579199	14.6%	4755	4208	11.5%	4496	3949	12.2%	0.24	0.54
fs900	1888992	1745834	8.2%	6000	5400	10.0%	5700	5100	10.5%	0.71	1.39
fs2116	fail	6208840	N/A	fail	8800	N/A	fail	8500	N/A	fail	9.46
fs4096	fail	16807614	N/A	fail	13300	N/A	fail	13000	N/A	fail	43.79
Average	10.2%			13.4%			13.9%				

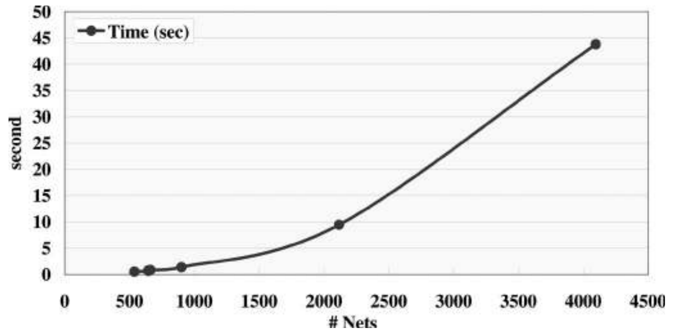


Fig. 21. Running time for the RDL routing.

TABLE V
MEMORY USAGE FOR EACH CIRCUIT

Circuits	#Nets	Memory (K bytes)
fwaa281aa	537	962
fsa0ac015aa	645	866
fs90b740	646	830
fsa0ac013aa	661	830
fs900	900	3258
fs2116	2116	6886
fs4096	4096	13555

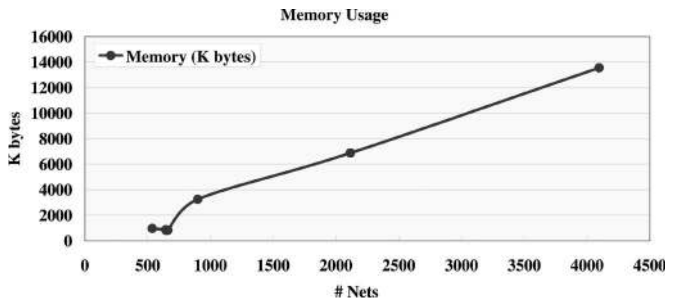


Fig. 22. Memory usage for the RDL routing.

and the signal skew may be further improved at the cost of larger total wirelength. Thus, we use this property to minimize the critical wirelength and the signal skew without increasing the total wirelength too much. We conducted an experiment to explore the effects of different α, and the results are listed in Table VI. In this experiment, we tested three pairs of the smallest α and the largest α values on fs90b740. We first set the largest (smallest) α to 1 and then scaled it up (down) to see the effects of different α to the total wirelength, the critical wirelength, and the signal skew. The percentages listed in the parentheses give the normalized ratios to that with the smallest and the largest α being set to 1. From the experimental results in Table VI, as the largest (smallest) α scales up (down), the total wirelength increases while the critical wirelength and the signal skew decrease.

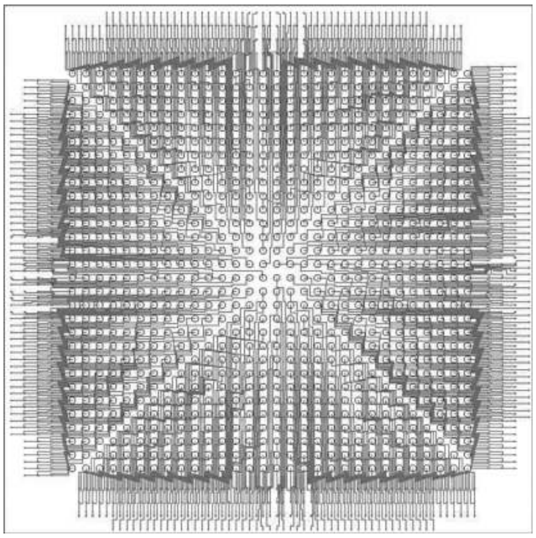


Fig. 23. RDL routing result for fs900.

V. CONCLUSION

In this paper, we have developed an RDL router for the flip-chip package. The RDL router consists of the two stages of global routing followed by detailed routing. The global routing applies the network flow algorithm to solve the assignment problem from the wire-bonding pads to the bump pads and then creates the global path for each net. The detailed routing applies the three-stage technique of cross-point assignment, net ordering determination, and track assignment to complete the routing. Experimental results demonstrate that our router can

TABLE VI
EFFECTS OF DIFFERENT α ON WIRELENGTH AND SKEW

α value / Result	Smallest α = 1, Largest α = 1	Smallest α = 2, Largest α = 0.2	Smallest α = 5, Largest α = 0.1
Total wirelength (μm)	776313 (100%)	779089 (100.4%)	808143 (104.1%)
Critical wirelength (μm)	3493 (100%)	3357 (96.1%)	3254 (93.2%)
Skew	3203 (100%)	3067 (95.8%)	2596 (81.0%)
CPU time (s)	0.68	0.68	0.68

achieve much better results in routability, wirelength, critical wirelength, and signal skews, as compared with a heuristic algorithm currently used in industry.

REFERENCES

[1] H. Cai, "Multi-pads, single layer power net routing in VLSI circuits," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 1998, pp. 183–188.

[2] D.-S. Chen and M. Sarrafzadeh, "A wire-length minimization algorithm for single-layer layouts," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1992, pp. 390–393.

[3] S.-S. Chen, J.-J. Chen, S.-J. Chen, and C.-C. Tsai, "An automatic router for the pin grid array package," in *Proc. ACM/IEEE Asia and South Pac. Des. Autom. Conf.*, Jan. 1999, pp. 133–136.

[4] S.-S. Chen, J.-J. Chen, C.-C. Tsai, and S.-J. Chen, "An even wiring approach to the ball grid array package routing," in *Proc. IEEE Int. Conf. Comput. Des.*, Oct. 1999, pp. 303–306.

[5] B. V. Cherkassky, "Efficient algorithms for the maximum flow problem," *Math. Methods Solut. Econ. Probl.*, vol. 7, pp. 117–126, 1977.

[6] J.-W. Fang, I.-J. Lin, P.-H. Yuh, Y.-W. Chang, and J.-H. Wang, "A routing algorithm for flip-chip design," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 2005, pp. 753–758.

[7] C.-P. Hsu, "General river routing algorithm," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 1983, pp. 578–583.

[8] K.-F. Liao, M. Sarrafzadeh, and C. K. Wong, "Single-layer global routing," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 13, no. 1, pp. 38–47, Jan. 1994.

[9] A. Titus, B. Jaiswal, T. J. Dishongh, and A. N. Cartwright, "Innovative circuit board level routing designs for BGA packages," *IEEE Trans. Adv. Packag.*, vol. 27, no. 4, pp. 630–639, Nov. 2004.

[10] C.-C. Tsai, C.-M. Wang, and S.-J. Chen, "News: A net-even-wiring system for the routing on a multilayer PGA package," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 2, pp. 182–189, Feb. 1998.

[11] UMC, *0.13 μm Flip-Chip Layout Guideline*, p. 6, 2004.

[12] D. Wang, P. Zhang, C.-K. Cheng, and A. Sen, "A performance-driven I/O pin routing algorithm," in *Proc. ACM/IEEE Asia and South Pac. Des. Autom. Conf.*, Jan. 1999, pp. 129–132.

[13] M.-F. Yu and W. W.-M. Dai, "Pin assignment and routing on a single-layer pin grid array," in *Proc. ACM/IEEE Asia and South Pac. Des. Autom. Conf.*, Sep. 1995, pp. 203–208.

[14] M.-F. Yu and W. W.-M. Dai, "Single-layer fanout routing and routability analysis for ball grid arrays," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1995, pp. 581–586.

[15] M.-F. Yu, J. Darnauer, and W. W.-M. Dai, "Interchangeable pin routing with application to package layout," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, Nov. 1996, pp. 668–673.

Jia-Wei Fang (S'05) received the B.S. degree in electrical engineering from the National Cheng Kung University, Tainan, Taiwan, R.O.C., in 2003 and the M.S. degree in electronics engineering from the National Taiwan University, Taipei, Taiwan, in 2005. He is currently working toward the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University.

His current research interests include package routing and other package technologies.

I-Jye Lin (S'05) received the B.S. degree in computer science from the National Tsing-Hua University, Hsinchu, Taiwan, R.O.C., in 2004. She is currently working toward the Ph.D. degree at the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan.

Her current research interests include package routing and design for manufacturing.

Yao-Wen Chang (S'94–M'96) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin in 1993 and 1996, respectively, all in computer science.

He is a Professor in the Department of Electrical Engineering and the Graduate Institute of Electronics Engineering, National Taiwan University. He is currently also a Visiting Professor at Waseda University, Japan. He was with the IBM T. J. Watson Research Center, Yorktown Heights, NY, in the summer of 1994. From 1996 to 2001, he was on the faculty of National Chiao Tung University, Taiwan. His current research interests lie in VLSI physical design, design for manufacturing, and FPGA. He has been working closely with industry on projects in these areas. He has coauthored one book on routing and over 120 ACM/IEEE conference/journal papers in these areas.

Dr. Chang received an award at the 2006 ACM ISPD Placement Contest, Best Paper Award at ICCD-1995, and nine Best Paper Award Nominations from DAC-2007, ISPD-2007 (two), DAC-2005, 2004 ACM TODAES, ASP-DAC-2003, ICCAD-2002, ICCD-2001, and DAC-2000. He has received many awards for research performance, such as the inaugural First-Class Principal Investigator Awards and the 2004 Mr. Wu Ta You Memorial Award from the National Science Council of Taiwan, the 2004 MXIC Young Chair Professorship from the MXIC Corp, and for excellent teaching from National Taiwan University and National Chiao Tung University. He is an editor of the *Journal of Computer and Information Science*. He has served on the ACM/SIGDA Physical Design Technical Committee and the technical program committees of ASP-DAC (topic chair), DAC, DATE, FPT (program co-chair), GLSVLSI, ICCAD, ICCD, IECON (topic chair), ISPD, SOCC (topic chair), TENCON, and VLSI-DAT (topic chair). He is currently an independent board member of Genesys Logic, Inc, the chair of the Design Automation and Testing (DAT) Consortium of the Ministry of Education, Taiwan, a member of the board of governors of the Taiwan IC Design Society, and a member of the IEEE Circuits and Systems Society, ACM, and ACM/SIGDA.

Jyh-Herng Wang (M'98) received the B.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1987 and 1994, respectively.

From 1994 to 1999, he was an Associate Scientist with the electrical design group, National Center for High-Performance Computing, HsinChu, Taiwan. He then joined Faraday Technology Corporation, where he was a Staff Engineer, CAD Manager, and Director of the Design Development Division, in 1999, 2000, and 2003, respectively. He joined Apache Design Solutions, Hsinchu, Taiwan, as Director of the Taiwan R&D team in 2007. His research interests include timing analysis, SI analysis, and VLSI design flow.