A Routing Algorithm for Flip-Chip Design

Jia-Wei Fang[1], I-Jye Lin[2], Ping-Hung Yuh[3], Yao-Wen Chang[1,2], and Jyh-Herng Wang[4]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan
[2]Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan
[3]Department of Computer Science and Information Engineering, National Taiwan University, Taipei 106, Taiwan
[4]Faraday Technology Corporation, Hsinchu 300, Taiwan

***Abstract*— The flip-chip package gives the highest chip density of any packaging method to support the pad-limited Application-Specific Integrated Circuit (ASIC) 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 routing path for each net. The detailed routing consists of three stages, cross point assignment, net ordering determination, and 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%, compared with a heuristic algorithm currently used in industry.**

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, the *flip-chip (FC) package*, as shown in Figure 1, is created for higher integration density and rising power consumption. Flip-chip bonding was first developed by IBM in 1960's. It gives the highest chip density of any packaging method to support the pad-limited ASIC designs.

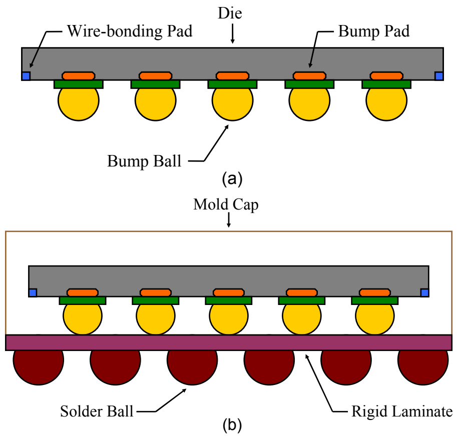


Fig. 1. (a) A Flip Chip. (b) A Flip Chip Package.

Flip-chip is not a specific package, or even a package type (like PGA or BGA). Flip-chip describes the method of electrically connecting the die to the package carrier. The package carrier, either a substrate or a leadframe, provides the connection from the die to the outside devices of the package. The die 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 IC designs, the I/O pads are still placed along the boundary of the die. This placement does not suit for the flip-chip package. As a result, we use the top metal or an extra metal layer, called a *Re-Distributed Layer (RDL)* as shown in Figure 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. Bump balls are placed on the RDL and use the RDL to connect to wire-bonding pads by bump pads.

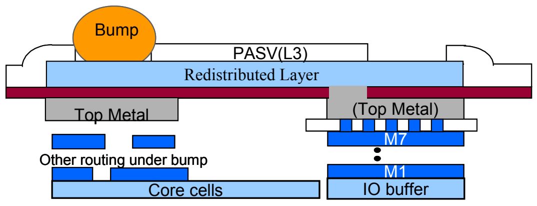


Fig. 2. Cross Section of RDL

The flip-chip package is generally classified into two types: the *peripheral array* as shown in Figure 3(a) and the *area array* as shown in Figure 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 the limited number of bump balls. In the area array, 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, so 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, the *Redistribution Layer (RDL) router* [13], is needed to reroute the peripheral wire-bonding pads to the bump pads and then connect the bump pads to the bump balls. Considering the routing of multi-pin nets and the minimization of total wirelength and the signal skews are also needed for an RDL router. Figure 3(c) shows one RDL routing result for an area-array flip-chip.

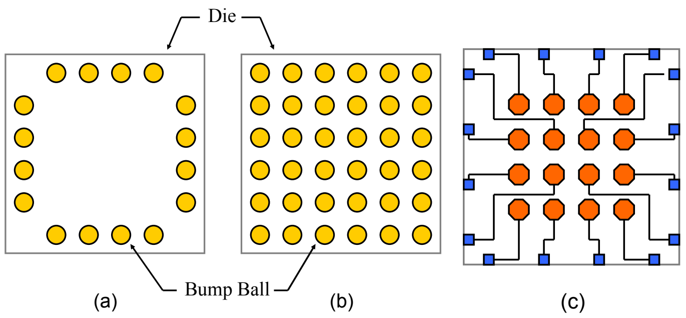


Fig. 3. (a) A Peripheral Array. (b) An Area Array. (c) An RDL Routing Result.

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 ball grid array (BGA) packages and pin grid array (PGA) packages, including [3], [10], [11], [12], [14], [16] and [17]. The work [16] used the geometric and symmetric attributes of the pin positions in the BGA packages to assign pins of the BGA. However, in flip-chip designs the positions of wire-bonding pads and bump pads do not always have these geometric and symmetric attributes. The works [3] and [11] presented PGA routers while [12] provided a BGA router. These three routers are any-angle, multi-layer routers without considering the pin assignment problem, single-layer routing, and total wirelength minimization. The works [14] and [17] 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 multi-pin nets and signal skews. The work [14] also did not consider the routing congestion problem. Furthermore, they assumed that wires can be any-angle, so their methods are not suitable for the RDL routing, typically with 90-degree angle routing.

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 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 algorithm, we can guarantee 100% detailed routing completion after the assignment. The second phase is the detail routing that efficiently distributes the routing points between two 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%, compared with a heuristic algorithm currently used in industry.

The rest of this paper is organized as follows. Section 2 gives the formulation of the RDL routing problem. Section 3 details our global and detailed routing algorithms. Section 4 shows the experimental results. Finally, conclusions are given in section 5.

II. Problem Formulation

We introduce the notations used in this paper and formally define the routing problem for flip-chip package. Figure 4 shows the modeling of the routing structure of the flip-chip package. Let P be the set of wire-bonding pads, and 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 $R_b = \{r_1^b, r_2^b, .., r_m^b\}$ be a set of m bump pad rings in the center of the package, and let $R_p = \{r_1^p, r_2^p, .., r_k^p\}$ 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 $\{b_1^i, b_2^i, .., b_q^i\}$, and each wire-bonding pad r_j^p consists of l wire-bonding pads $\{p_1^j, p_2^j, .., p_l^j\}$. Let N be the set of nets for routing. Each net n in N is defined by a set of wire-bonding pads and a set of bump pads that should be connected. Thus n can be a multi-pin net. 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 Figure 4, based on the two diagonals of the flip-chip package, we partition the whole package into four sectors: $North = \{P_N, B_N, R_p^N, R_b^N\}$, $East = \{P_E, B_E, R_p^E, R_b^E\}$, $South = \{P_S, B_S, R_p^S, R_b^S\}$, and $West = \{P_W, B_W, R_p^W, R_b^W\}$, 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, the *monotonic routing* and the *non-monotonic routing*. A monotonic routing can be formally defined as the 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 showing in Figure 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 routing of n_2 and n_4 are non-monotonic routing as shown in Figure 5(b). A good flip-chip package routing should be a monotonic routing because the monotonic routing results in smaller total wirelength and higher routing completion, compared to the non-monotonic routing.

Based on the definition above, the routing problem can be formally defined as the 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 and the routing is monotonic, the total wirelength is minimized, and the signal skew is minimized.

III. The 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.

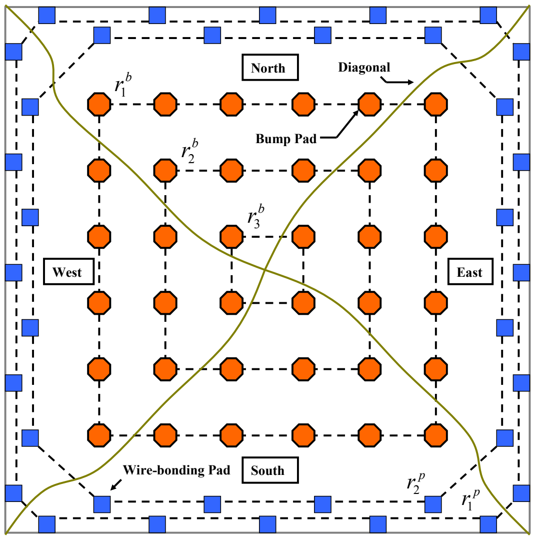


Fig. 4. Four Sectors in a Flip-Chip Package.

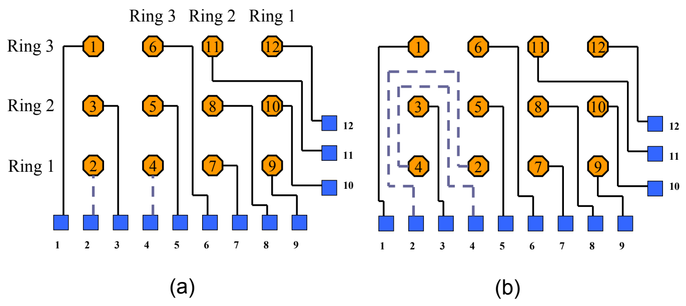


Fig. 5. (a) Monotonic Routing. (b) Non-monotonic Routing.

A. Algorithm Overview

According to the routing flow shown in Figure 6, our algorithm consists of two phases: (1) global routing based on the *minimum-cost maximum-flow (MCMF) algorithm* [2], 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 G_N, G_E, G_S, and 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 2-pin nets and then multi-pin nets. The reason is that 2-pin nets have less freedom to choose the routing path, so it needs to be considered first. 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 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 [6] 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. Figure 7 summaries our routing algorithm.

B. Global Routing

In this subsection, 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 multi-pin nets.

1) Basic Network Formulation: We describe how to construct the flow network G_S to perform the assignment for the $South$ sector. The other three sectors can be processed similarly. As shown in Figure 8(a), we define $D_S = \{d_1^S, d_2^S, .., d_h^S\}$ to be a set of h *intermediate nodes*.

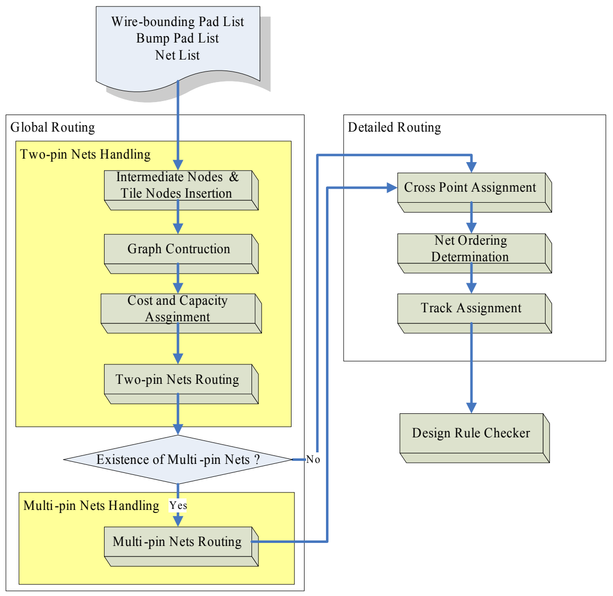


Fig. 6. The RDL Routing Flow

```
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 $b \in B$
5          in the same sector and the global routing 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 $b \in B$
10         in the same sector and the global routing 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
```

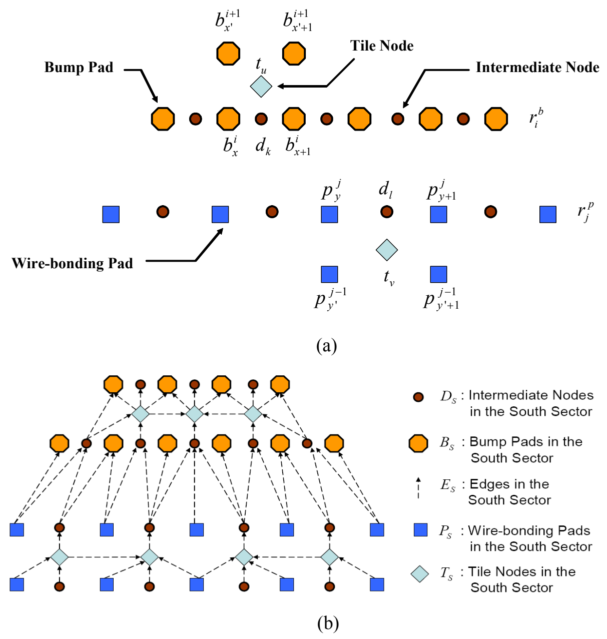
Fig. 7. Overview of the RDL Routing Algorithm.

Each intermediate node represents an interval (b_x^i, b_{x+1}^i) $((p_y^j, p_{y+1}^j))$ in a bump pad ring (wire-bonding pad ring). $T_S = \{t_1^S, t_2^S, .., t_u^S\}$ is a set of u *tile nodes*. Each tile node represents a tile $(b_x^i, b_{x+1}^i, b_{x'}^{i+1}, b_{x'+1}^{i+1})$ $((p_y^j, p_{y+1}^j, p_{y'}^{j+1}, p_{y'+1}^{j+1}))$ between two adjacent bump pad rings (wire-bonding 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 target node t to G_S. Each intermediate node has a capacity K, where K represents the maximum number of nets that are allowed to pass through an interval. Each tile node has a capacity L, where L represents the maximum number of nets that are allowed to pass through a tile. 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-B.2. There are nine 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 an intermediate node to a bump pad,
4) edges from an intermediate node to another intermediate node,
5) edges from an intermediate node to a tile node,
6) edges from a wire-bonding pad to a tile node,
7) edges from a tile node to a bump pad,
8) edges from a tile node to an intermediate node, and
9) edges from a tile node to another tile node.

There is an edge from the source s to every node in P_S, and there is an edge from every node in B_S to the target t. Each edge is associated with a $(cost, capacity)$ tuple to be described in the following subsections. 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 not any crossings in edges. Thus, we construct all the edges and avoid crossings of all edges at the same time. Figure 8(b) shows an example flow network G_S for the *South* sector. We can solve MCMF in time $O(|V|^2|E|^{\frac{1}{2}})$ [2], 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 $O(|V|^2|E|^{\frac{1}{2}})$ time.



$b_{x'}^{i+1}$
$b_{x'+1}^{i+1}$
Tile Node
Bump Pad
t_u
Intermediate Node
r_i^b
b_x^i
d_k
b_{x+1}^i
p_y^j
d_l
p_{y+1}^j
r_j^p
Wire-bonding Pad
t_v
$p_{y'}^{j-1}$
$p_{y'+1}^{j-1}$
(a)
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
(b)

Fig. 8. (a) Intermediate Nodes and Tile Nodes. (b) Flow Network for the South Sector.

2) Capacity Assignment and Node Construction: Now we introduce the capacity of each edge, the intermediate nodes, and the tile nodes. 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 1. If e is from an intermediate node or a tile node to a bump pad, then the capacity of e is set to M, where M is the maximum number of nets that are allowed to connect to the bump pad. Recall that an intermediate node has the capacity of K, where K is the maximum number of nets that are allowed to pass through this intermediate node. This means that the number of all outgoing edges of an intermediate node d is equal to K. The same condition holds for all incoming edges of d. If e is from a tile node to another tile node, then the capacity of e is set to L, where L is the maximum number of nets that are allowed to pass through the tile node. As shown in Figure 9, 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 capacity K. All outgoing edges of d are now connected from d' with capacity K, and all incoming edges of d are now connected to d'' with capacity K. A tile node is also decomposed into two tile nodes t' and t'', and the capacity of a tile node is set to L, where L is the maximum number of nets that are allowed to pass through this tile node. The capacity of the edges from the source node to the wire-bonding pads is set to 1, and the capacity of the edges from the bump pads to the sink node is set to M. There are three worst cases of congestion in a tile, as shown in Figure 10. The four nodes in the three figures are all bump pads. In Figures 10(a) and (b), the maximum number of nets passing through the tile is $2K$. In Figure 10(c), the maximum number of nets passing through the tile is $3K$. If we do not use the tile node, the maximum number of nets in Figures 10(a), (b), and (c) could exceed the capacity of a tile ($2K > L$ or $3K > L$). Since the capacity of each tile node is well modeled in our flow network, we can totally avoid this congestion problem.

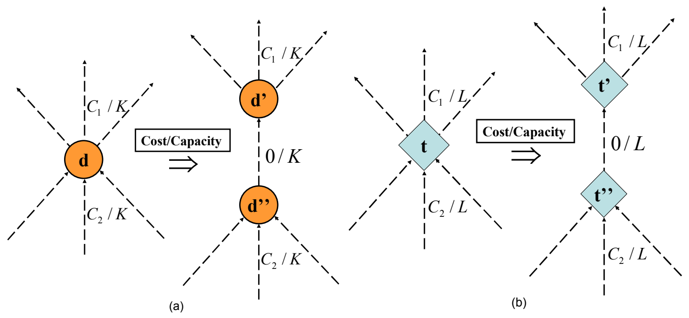


Fig. 9. (a) Capacity and Cost on Intermediate Nodes. (b) Capacity and Cost on Tile Nodes.

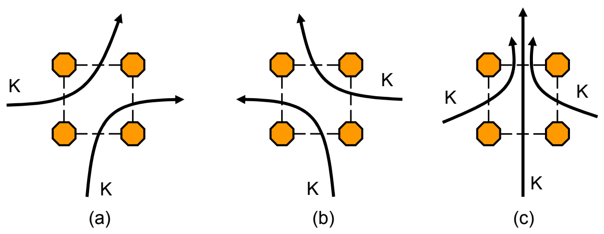


Fig. 10. Three Kinds of Congestion in a Tile.

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

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

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. We assign the smallest α to the edge that connects an intermediate node and a bump pad to assure that the intermediate nodes are assigned to bump pads first. The edge which connects two tile nodes are also assigned the smallest α to assure that fewer bump pad rings are used. The edge which connects a tile node to a bump pad or an intermediate node to a tile node is assigned a medium α. The edge that connects two intermediate nodes are assigned the largest α. By adjusting the value of α, we can control the wirelength of each net to avoid large signal skews among different nets. 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 0. Figure 11 shows the capacity and cost for all eight types of edges.

4) Multi-pin Net Handling: Finally, we describe how to deal with multi-pin nets. As stated before, we first assign 2-pin nets and then multi-pin nets. We only construct the edges associated with the 2-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 target node t. The global paths of the 2-pin nets are not deleted and considered as blockages F during the construction of the edges for the multi-pin 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 multi-pin nets, an edge e exists only if e does not intersect any blockages. Then we add the edges from the source node to the wire-bonding pads associated with the multi-pin nets and the edges from the bump pads associated with the multi-pin nets to the target node. Figure 12 illustrates an example. Assume that a multi-pin net n consists of $((p_2, p_4, p_5), (b_3, b_9))$, which means that p_2, p_4, and p_5 are free to be assigned to one of the two bump pads b_3 and b_9. 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 $\{p_4, p_5, b_9\}$.

In our global routing stage, the MCMF is optimal for two pin nets and suboptimal for multi-pin nets. Since we 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 flow network construction, the final routing solution will not generate wire crossings. So after the assignment, all global paths are routable. Based on above discussions, 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.

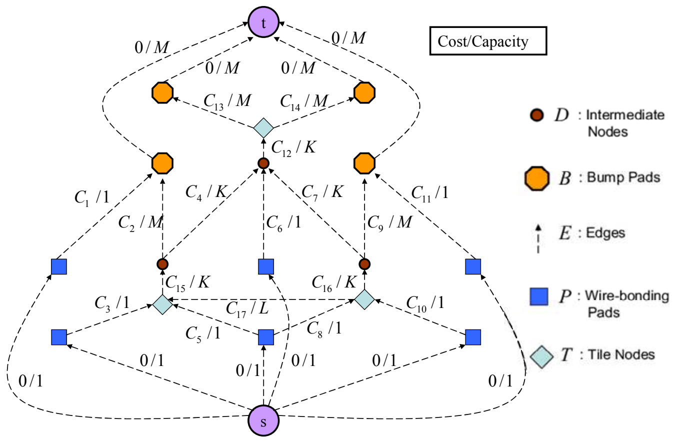


Fig. 11. Capacity and Cost on Edges.

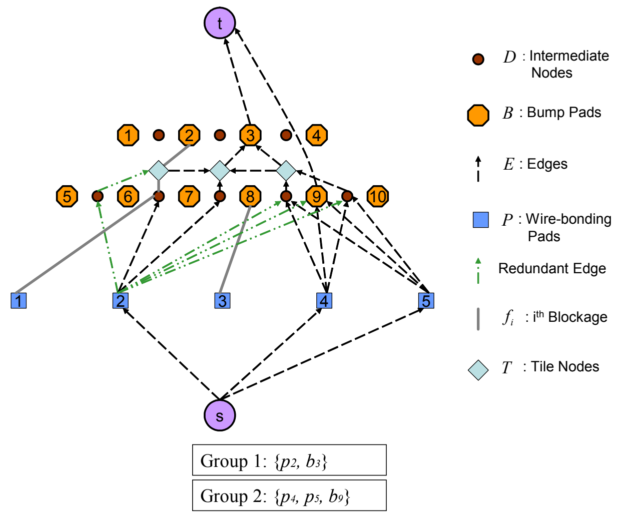


Fig. 12. Group Multi-pin Nets.

C. Detailed Routing

In this subsection, we explain the three methods used in our detailed routing. As shown in Figure 13, after the global routing, each global path contains only wire-bonding pads, intermediate nodes, and bump pads. The two global paths $< d_k, t, d_l >$ and $< d_k, t, b_x >$ which pass through the tile node t are remodelled as $< d_k, d_l >$ and $< d_k, b_x >$. Tile nodes are not needed for the final representations of the global routing paths because a tile node is just used to avoid the congestion overflow.

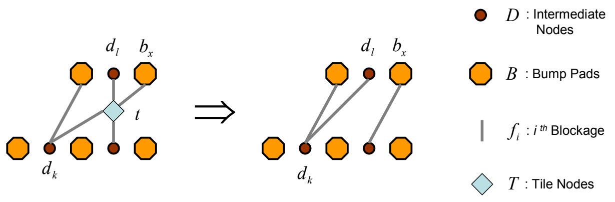


Fig. 13. Redefined Global Paths.

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 Figure 14 as an example. As shown in Figure 14, the two nets from wire-bonding pads p_2 and p_3 pass through the same intermediate node. So we split the intermediate node into two cross points.

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

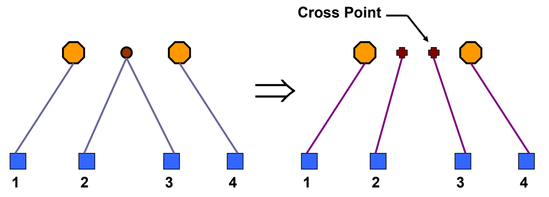
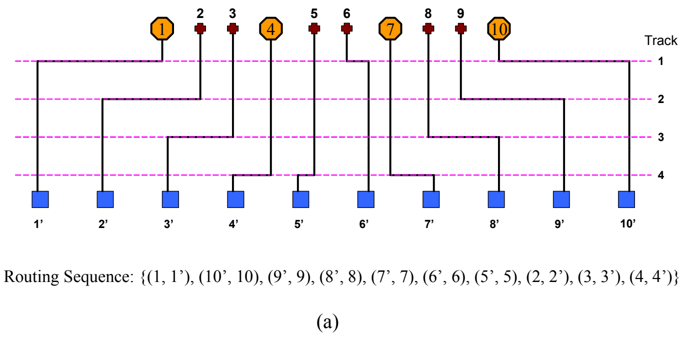


Fig. 14. Cross Point Assignment.

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. So we can use the net ordering determination algorithm presented in [6] to generate a routing sequence $S = <(n^s_1, n^t_1), (n^s_2, n^t_2), .., (n^s_k, n^t_k)>$ with k net segments. Each net segment n_j is represented by a pair $(source, target) = (n^s_j, n^t_j)$. We first determine the source and target for each net based on the counterclockwise traversing distance along the leftmost and the rightmost boundaries. For example, given the net 1 shown in Figure 15(a), since the distance along the leftmost boundary is smaller than the distance along the rightmost boundary, we make the terminal 1 a source and the terminal $1'$ a target. Starting from an arbitrary terminal, we then generate a circular list for all terminals ordered counter-clockwise 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 n_i, if it is a source, then we push it into the stack. Otherwise, if this terminal is a target and the top element of the stack belong to the same net, then net n_i is matched and the top element is popped. We keep searching the circular list until all nets are matched. 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 Figure 15(a), according to the net ordering determination algorithm described above, we can obtain the sequence $S = <(n_1, n'_1), (n'_{10}, n_{10}), (n'_9, n_9), (n'_8, n_8), (n'_7, n_7), (n'_6, n_6), (n'_5, n_5), (n_2, n'_2), (n_3, n'_3), (n_4, n'_4)>$.

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


1
2
3
4
5
6
7
8
9
10
Track
1
2
3
4
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)

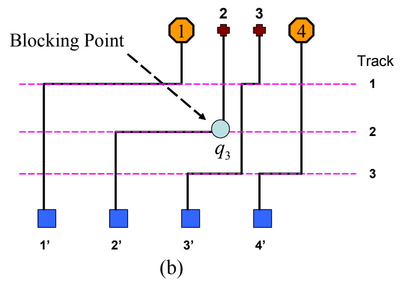


Fig. 15. (a) An Example for Track Assignment. (b) Blocking Point.

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. (For example, for a small test case with 513 nets, we need 25 minutes on a 1.2GHz SUN Blade 2000 workstation with 8 GB memory to complete the detailed routing.) So 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^s_i and n^t_i, 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^s_i is on the top-right side of n^t_i, or n^s_i is on the bottom-right side of n^t_i. Otherwise, we search the tracks from the bottom to the top. If we find a track l and it does not create any overlap with other wires, then we assign l to n_i. As shown in Figure 15(a), n_1 is assigned to track 1 first, and n_5 is assigned to track 4 first. Also we record the blocking points Q for n_i. A *blocking segment* is a wire on track $l+1$ (if we search from the top to the bottom) or $l-1$ (if we search from the bottom to the top) to stop n_i from being assigned to $l+1$ or $l-1$ without creating any overlap with it. A *blocking point* q_i is a terminal of the blocking segment whose projection on l overlaps with n_i. As shown in Figure 15(b), the point q_3 on track l_2 is the blocking point for net n_3. If we cannot find such l, we rip-up and reroute all net segments n_1 to n_{i-1}. For each net 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 Figure 15(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 bottom) 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. Figure 16 summarizes the track assignment algorithm.

```
Algorithm: Track Assignment($S_j$, $L$)
$S_j$: a routing sequence between rings $r_j$ and $r_{j+1}$;
$L$: the maximum number of tracks;
1 begin
2    for each net segment $n_i$ in $S_j$
3       Let $(x^s_i, y^s_i)$ $((x^t_i, y^t_i))$ be the coordinate of the
4          source (target) of $n_i$;
5       if $((x^s_i \ge x^t_i$ and $y^s_i \ge y^t_i)$ or $(x^s_i \ge x^t_i$ and $y^s_i \le y^t_i))$
6          Find a track $l$ of $L$ from the top to the bottom without
7             creating an overlap with other wires;
8       else
9          Find a track $l$ of $L$ from the bottom to the top without
10            creating an overlap with other wires;
11      if such $l$ exists
12         Assign $l$ to $n_i$;
13      else
14         for all pre-routed net $n_k$
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 current search
19               direction (from top to bottom or bottom to top);
20 end
```

Fig. 16. Algorithm for Track Assignment.

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

IV. EXPERIMENTAL RESULTS

We implemented our algorithm in the C++ programming language on a 1.2GHz SUN Blade 2000 workstation with 8 GB memory. The benchmark circuits fs90b740, fsa0ac013aa, fsa0ac015aa, fwaa281, fs900, fs2116, and fs4096 are real industry designs.

TABLE I
TEST CASES FOR RDL ROUTER.

Case name	# 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

In Table I, “Case name” denotes the names of circuits, “#Nets” denotes the number of nets, “#R_p” denotes the number of wire-bonding pad rings,

TABLE II
RDL ROUTING RESULTS.

Algorithm / Case name	Total wirelength (μm)			Critical wirelength (μm)			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%**				

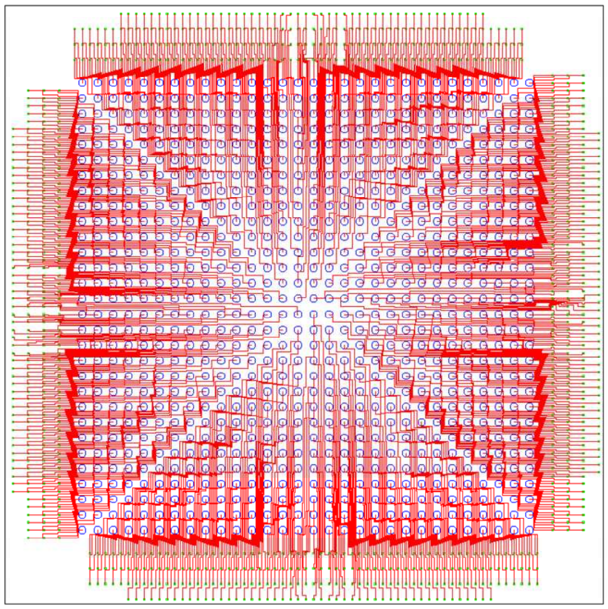


Fig. 17. RDL Routing Solution of fs900.

"$\#p$" 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. So 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.

Since there are no flip-chip routing algorithms in the literature, we compared our algorithm with the following heuristic algorithm 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 II. We report the total wirelength, the critical wirelength, the maximum signal skews, and the CPU times. Since the routability is guaranteed to be 100%, we do not report it. 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. Figure 17 shows the RDL routing result of fs900. The experimental results demonstrates the effectiveness of our network flow based algorithm for the routing for flip-chip designs.

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 routing 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 show that our router can achieve much better results in routability, wirelength, critical wirelength, and signal skews, compared with a heuristic algorithm currently used in industry.

REFERENCES

[1] D. Chang, T. F. Gonzalez, and O. H. Ibarra, "A Flow Based Approach to the Pin Redistribution Problem for Multi-Chip Modules," *Proc. GVLSI*, pp. 114–119, 1994.

[2] B. Cherkasssky, "Efficient Algorithms for the Maximum Flow Problem," *Mathematical Methods for the Solution of Economical Problems, vol. 7*, pp. 117–126, 1977.

[3] S.-S. Chen, J.-J. Chen, S.-J. Chen, and C.-C. Tsai, "An Automatic Router for the Pin Grid Array Package," *Proc. ASP-DAC*, pp. 133–136, 1999.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 2000.

[5] M.-F. Yu and W.-M. Dai, "Pin Assignment and Routing on a Single-Layer Pin Grid Array," *Proc. ASPDAC*, pp. 203–208, 1995.

[6] C.-P. Hsu , "General River Routing Algorithm," *Proc. DAC*, pp. 578–583, 1983.

[7] J. Hu and S. S. Sapatnekar, "A Timing-Constrained Algorithm for Simultaneous Global Routing of Multiple Nets," *Proc. ICCAD*, pp. 99–103, 2000.

[8] Y. Kubo and A. Takahashi, "A Global Routing Method for 2-Layer Ball Grid Array Packages," *Proc. ISPD*, pp. 36–43, 2005.

[9] E. S. Kuh, T. K. Kashiwabara, and T. Fujisawa, "On Optimum Single Row Routing," *IEEE Transactions on Circuits and Systems, vol. 26*, pp. 361–368, 1979.

[10] A. Titus, B. Jaiswal, T. J. Dishongh, and A. N. Cartwright, "Innovative Circuit Board Level Routing Designs for BGA Packages," *IEEE Transactions on Advanced Packaging, vol. 27*, pp. 630–639, 2004.

[11] 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 Transactions on Computer-Aided Design, vol. 17*, pp. 182–189, 1998.

[12] S.-S. Chen, J.-J. Chen, C.-C. Tsai, and S.-J. Chen, "An Even Wiring Approach to the Ball Grid Array Package Routing," *Proc. ICCD*, pp. 303–306, 1999.

[13] UMC, "0.13μm Flip Chip Layout Guideline," pp. 6, 2004.

[14] D. Wang, P. Zhang, C.-K. Chang, and A. Sen, "A Performance-Driven I/O Pin Routing Algorithm," *Proc. ASP-DAC*, pp. 129–132, 1999.

[15] X. Xiang, X. Tang, and D.-F. Wang, "Minimum-Cost Flow-Based Algorithm for Simultaneous Pin Assignment and Routing," *IEEE Transactions on Computer-Aided Design, vol. 22*, pp. 870–878, 2003.

[16] M.-F. Yu and W.-M. Dai, "Single-Layer Fanout Routing and Routability Analysis for Ball Grid Arrays," *Proc. ICCAD*, pp. 581–586, 1995.

[17] M.-F. Yu, J. Darnauer and W.-M. Dai, "Interchangeable Pin Rouing with Application to Package Layout," *Proc. ICCAD*, pp. 668–673, 1996.