Flip-Chip Routing with Unified Area-I/O Pad Assignments for Package-Board Co-Design *

Jia-Wei Fang[1,2], Martin D. F. Wong[2], and Yao-Wen Chang[1,3]
[1]Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
[2]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA
[3]Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

ABSTRACT

In this paper, we present a novel flip-chip routing algorithm for package-board co-design. Unlike the previous works that can consider only either free- or pre-assignment routing, our router is the first work in the literature that can handle both the free- and pre-assignment routing. Based on the computational geometry techniques (e.g., the Delaunay triangulation and the Voronoi diagram), the router applies a unified network-flow formulation to perform congestion estimation for the pre-assignment routing. According to the congestion map, the network-flow formulation can also consider the free-assignment nets during the routing for the pre-assignment ones. Then, the router modifies the network-flow formulation to optimally assign and route the free-assignment nets, considering the routed pre-assignment nets. With the package and board co-design flow, we can achieve 100% routing completion. Experimental results based on industry designs demonstrate the high-quality of our algorithm.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design Aids - Layout, Placement and Routing
General Terms: Algorithms, Design
Keywords: Detailed Routing, Global Routing, Physical Design

1. INTRODUCTION

In VLSI designs, the increasing complexity and decreasing feature size have made the demand of more I/Os a significant problem to packaging technologies. An advanced packaging technology, the *flip-chip package*, as shown in Figure 1(a), is created for higher integration density and larger I/O counts. In recent IC's, designers place the I/O pads (buffers) in the whole area of a die, instead of just along the die boundary, to result in shorter wirelength, higher chip density, and better signal and power integrity.

For the flip-chip applications, typically the top metal or an extra metal layer, called a *re-distribution layer* (*RDL*) as illustrated in Figure 1(b), is used to redistribute/connect the *I/O pads* to the *bump pads* without changing the placement of the I/O pads [5].

There are two kinds of the RDL routing problems for the flip-chip design. The first one is the *free-assignment* (*FA* for short) routing problem. In this problem, a router has the freedom to assign each I/O pad with a signal to any bump pad with no signal during routing [3]. Therefore, it ignores the routing constraints from a printed circuit board (PCB). The second kind of RDL routing is the *pre-assignment* (*PA* for short) routing problem where the mapping among the I/O pads and the bump pads is pre-defined before routing and cannot be changed [2]. Further, the pre-defined netlist can provide the routing constraints from the PCB. However, if the netlist generation does not consider the package routing constraints, the RDL routing might not be completed. Hence, for package-board (PB) co-design, the two routing problems shall be considered simultaneously to achieve greater design flexibility and better design performance. Figure 2 shows two routing examples. An I/O pad and a bump pad with the same label i form a PA net i, e.g., nets 1–4. Other I/O pads which are not paired with any bump pad are the FA nets, e.g., nets 5–8. In Figure 2(a), since the routing does not consider PB co-design, net 2 might block net 7, and net 8 forces net 1 to detour. In contrast, in Figure 2(b), considering PB co-design gives a feasible routing for net 7 and further reduces the total wirelength. Therefore, it is desirable to develop an RDL routing algorithm to consider both the PA and FA nets for PB co-design.

*This work was partially supported by ITRT, SpringSoft, Synopsys, TSMC, and National Science Council of Taiwan under Grant No's. NSC 97-2221-E-002-237-MY3, NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, and NSC 096-2917-I-002-120.

DAC'09, July 26–31, 2009, San Francisco, California, USA.

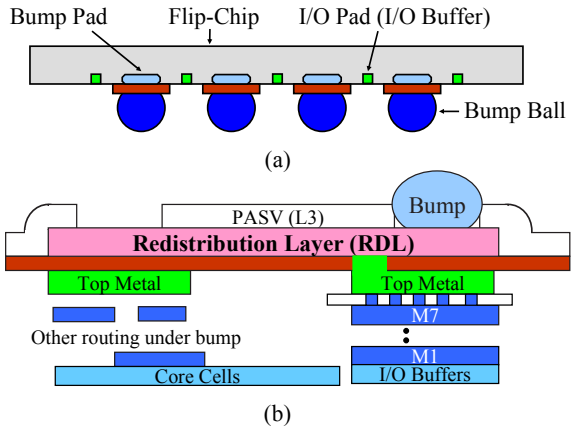

Figure 1: (a) Flip-Chip Package. (b) Section of RDL.

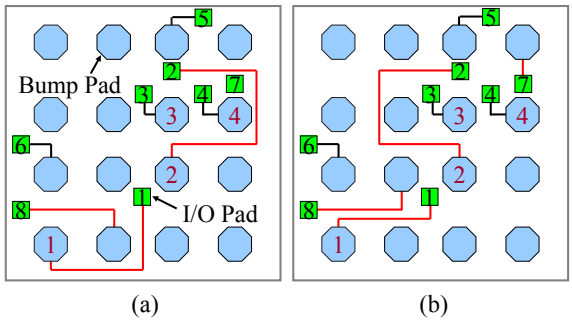

Figure 2: (a) Routing without Package-Board Co-Design. (b) Routing with Package-Board Co-Design.

1.1 Previous Work

To the best of our knowledge, there is no existing work in the literature specially for the area-I/O flip-chip routing that consider both the PA and FA nets for PB co-design. Two related works on flip-chip routing only consider the FA routing problem [3] or the PA routing problem [2], but not both together. Furthermore, the work [2] can only handle the flip-chip structure with the I/O

pads placed along the die boundary. Although the work [3] is for the area-I/O flip-chip structure, its *tile-based* flow-network construction did not consider the PA nets and the positions of I/O pads which can avoid over design. Another related work is the routing for Pin-Grid-Array (PGA) packages [6]. The work [6] used triangulation and its dual to assign pins in the PGA packages. Since the work represented pre-routed nets by edges (obstacles) in the triangulation (dual), the topology of the nets cannot be changed during routing. For PB co-design, however, FA and PA nets affect each other, and thus the changes of the topology are inevitable. In addition to the aforementioned differences, a major deficiency of the previous works is that they cannot handle the unified constraints induced by both the FA and PA routing.

1.2 Our Contributions

We present in this paper a novel flip-chip routing algorithm for package and board co-design. Unlike the previous works that can consider only either FA or PA routing, our router is the first work in the literature that can handle both the FA and PA routing. Based on the computational geometry techniques (e.g., the Delaunay triangulation [DT] and the Voronoi diagram [VD]), the global routing applies a unified network-flow formulation to perform congestion estimation for the PA routing. With the DT and VD techniques, the positions of I/O pads can be captured more precisely to avoid wire congestion. According to the congestion map, the network-flow formulation can also consider the FA nets during the routing for the PA ones. Then, the router modifies the network-flow formulation to optimally assign and route the FA nets, considering the routed PA nets. With the package and board co-design flow, we can achieve 100% routing completion. Experimental results based on industry designs demonstrate that our router can achieve 100% routability and shorter routed wirelength, compared with the related works.

The rest of this paper is organized as follows. Section 2 gives the formulation of the routing problem. Section 3 details our routing algorithm. Section 4 reports the experimental results. Finally, our conclusion is given in Section 5.

2. PROBLEM FORMULATION

We introduce the notation used in this paper and formally define the RDL routing problem with both the FA and PA routing constraints for the PB co-design. Figure 3 shows the modelling of the routing structure of the area-I/O flip-chip package. Let $N = N_f \cup N_p$ be the set of FA nets (N_f) and PA nets (N_p) for routing. $Q = Q_f \cup Q_p$ is the set of I/O pads where Q_f and Q_p are the I/O pads of the FA and PA nets, respectively. B is the set of bump pads. For practical applications, each I/O pad is paired with only one bump pad, and several I/O pads can be routed to the same bump pad to form a multi-pin net. Let l_i be the bump-pad column i, and let w_j be the bump-pad row j. Each bump pad column/row (l_i/w_j) consists of a set of m bump pads, and each bump pad is represented by $b_{i,j}$. A tile is constructed by four adjacent bump pads $(b_{i,j}, b_{i+1,j}, b_{i+1,j+1}, b_{i,j+1})$. Since the RDL routing is typically on a single layer, it does not allow *wire crossings*, for which two wires intersect each other in the RDL.

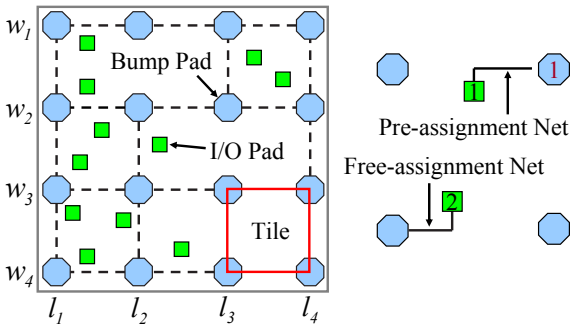

Figure 3: Routing Structure in a Flip-Chip Package.

Now we formally define the addressed problem as follows:

PROBLEM 1. *Given a netlist containing both free- and pre-assignment nets, and a placement of I/O pads and bump pads, the RDL routing problem for PB co-design is to connect a set of I/O pads and a set of bump pads so that there is no wire crossing in RDL and the total wirelength is minimized.*

3. THE ROUTING ALGORITHM

3.1 Algorithm Overview

In the routing flow of Figure 4, our algorithm consists of two major phases: (1) global routing based on DT, VD, and the *minimum-cost maximum-flow algorithm (MCMF)* [1], and (2) detailed routing based on routing-path refinement, net-ordering determination, and maze routing.

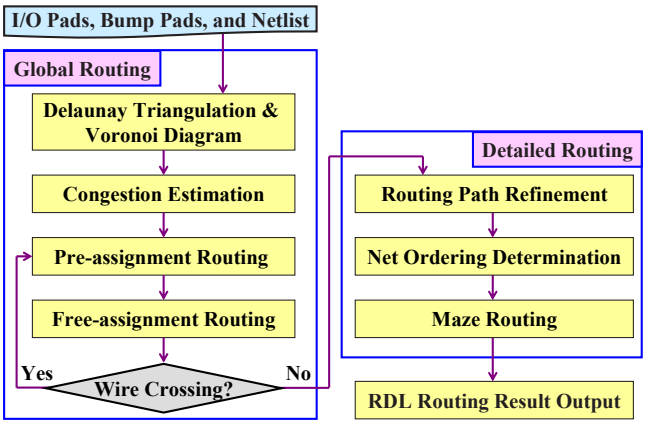

Figure 4: Routing Flow of Package-Board Co-design.

In the first phase, for PA nets, we construct the flow network G based on DT and VD to perform congestion estimation by MCMF. By the congestion map, we can apply maze routing to route the PA nets on G considering FA nets. Then we modify G to optimally assign and route the FA nets by MCMF based on the routed PA nets. To get better routing results, we allow the FA nets to cross the PA nets. If there are wire crossings, the crossed PA nets will be ripped up and rerouted or the crossed FA nets will be forbidden to choose the same routing paths.

In the second phase, we first separate every global-routing path from VD. Then we create a routing sequence in each tile that can guarantee to route all nets. Finally, we use maze routing to route each net based on the routing sequence.

3.2 Global Routing

3.2.1 Congestion Estimation

We construct the flow network G to perform the *concurrent assignment and routing* for the FA nets without considering the PA ones. Since this network-flow based routing can achieve the optimal results for the FA nets, the results can provide an accurate estimation of the congestion for routing the PA ones. Figure 5 shows an example where nets 1–4 are PA nets, and nets 5–8 are FA ones. In Figure 5(a), we first insert a set D of DT nodes on the die boundary and at the middle of each side of the bump pads. Based on the I/O pads and the DT nodes, we can obtain the corresponding DT (see the dashed edges). By the DT nodes, we can avoid edge crossings with the bump pads and use the routing space between the bump pads and the die boundary. Based on the DT, we can get VD and its corresponding edges (see the solid edges in Figure 5(a); for better illustration, only a partial VD is shown in the figure). The white nodes give the set M of VD nodes. They are inserted at each crossing of VD edges and the center of every FA bump pad. Figure 5(b) shows part of G. According to the VD, we construct $G = (Q \cup M \cup \{s,t\}, E)$, where E denotes the edge set, s is the source node, and t is the sink node. There are five types of edges:

1. Directed edge from an I/O pad to a VD node,
2. Bi-directional edge between a VD node and another one,
3. Directed edge from a VD node to a VD node in a bump pad,
4. Directed edge from the source node to an I/O pad,
5. Directed edge from a VD node in a bump pad to the sink node.

The last two types of edges are not shown in Figure 5. Type-1 edges are constructed from an I/O pad to its surrounding VD nodes. Type-2 edges are the VD edges. Type-3 edges are connected to a VD node in a bump pad from its surrounding VD nodes. Each edge is associated with a $(cost, capacity)$ ordered-pair. The cost is the length l_e of the edge e. The capacity is used to avoid wire congestion and equals the maximum number of wires allowed to pass through the edge. For each Type-2 edge (VD edge), its capacity is n_{max} which can be measured by the

length of the DT edge crossing the VD edge. The capacity of the other edges is 1 because only one signal can be transmitted from an I/O pad and to a bump pad. Since we also want to avoid wire congestion in every triangle, the capacity of each VD node is set to be the maximum capacity of all the edges connecting the VD node. Figure 5(c) shows an example routing result. Recall that we do not allow crossings for all wires. Since E represents the potential global-routing paths of all nets, we can guarantee that no wire crossing will occur if there exists no crossing among edges. As a result, we construct all edges and avoid their crossings at the same time. After applying MCMF, we can use the routing results for the FA nets to accurately estimate the congestion (see the arrows in Figure 5(a)) because MCMF gives the optimal routes (with the minimum wirelength). MCMF can be solved in time $O(|V|^2\sqrt{|E|})$ [1], where V is the vertex set.

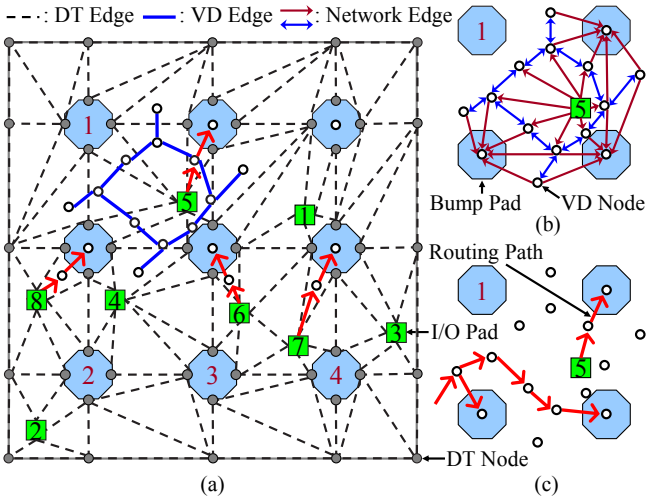

Figure 5: (a) DT and VD. (b) Flow Network. (c) Global-routing Paths of FA Nets.

3.2.2 Pre-Assignment Routing

According to the congestion estimation, some edges of G might be identified as congestion edges with routed FA wires (see Figure 6(a)). We first divide each multi-pin PA net into a set of 2-pin nets. Then we apply maze routing to route the PA nets on G. The new cost of each vertex v in G is $\alpha \times d_b$, where d_b is the central distance between two adjacent bump pads, and α is a user-defined positive constant if the vertex is a high-cost VD node which represents a crossing between two nets; $\alpha = 0$, otherwise. The new cost function of each edge e in G is defined by

$$Cost(e) = l_e + \beta \times d_b \times (n_e - n_{max} + 1), \quad (1)$$

where n_e is the number of routed wires through e, including the FA and PA ones. If $n_e - n_{max} + 1 > 0$, β is also a user-defined positive constant. Otherwise, $\beta = 0$. In Figure 6(b), if the PA routing path crosses the wires of I/O pad 5, there is an additional cost $\alpha \times d_b$ for passing the high-cost VD node. Once a cross is generated, we make the cost of the high-cost VD node zero in order not to count the cross again. The reason is that a crossed FA net can be removed to avoid the crossings. If the PA routing path is routed through the congestion edges (see the dotted ellipse), the cost of each edge will be computed by Equation 1. Figure 6(a) shows the routing result of the PA nets, where net 1 crosses net 5, and net 3 detours to avoid crossing net 7. Since the routing results for the FA nets are just used for congestion estimation, we allow wire crossings at this step. Note that no wire crossing is allowed between the PA nets.

3.2.3 Free-Assignment Routing

After the PA routing, we modify G to optimally route all FA nets based on the routed PA wires. In Figure 7(a), the routed PA wires become obstacles during routing the FA nets. Then we add extra VD nodes into G (see Figure 7(b)) to route FA nets along the PA wires and thus to avoid wire crossings. The extra VD nodes are inserted around the VD nodes on the routed PA wires. The number of extra VD nodes around a VD node equals the number of edges connecting the VD node. For example, there

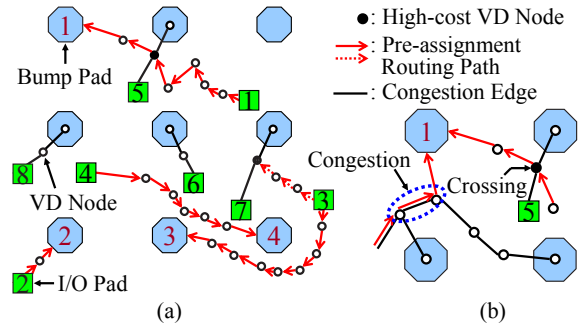

Figure 6: (a) PA Routing based on VD. (b) Additional Costs of Wire Crossings and Wire Congestion.

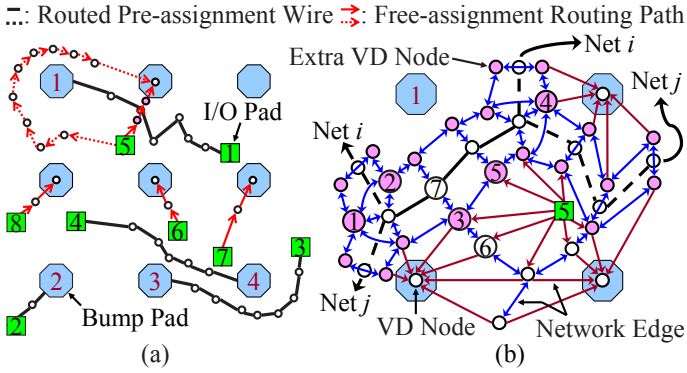

Figure 7: (a) FA Routing based on Modified Flow Network. (b) Modified Flow Network.

are three edges of the VD node connecting to extra VD node 1. Thus, we insert an extra VD node between two adjacent edges.

After inserting the extra VD nodes, we remove all the dashed edges and the edges connecting the routed PA wires from G to avoid wire crossings. We will construct additional edges later to replace the removed ones. The thick edges will be kept because the FA nets can be routed between nets i and j (extra VD nodes 1 and 4). Then in Figure 7(b), we have additional five types of edges in G (over the previous types in Section 3.2.1) as follows:
6. Directed edge from an I/O pad to an extra VD node,
7. Bi-directional edge between an extra VD node and another one along a VD edge,
8. Bi-directional edge between an extra VD node and another one around the same VD node,
9. Bi-directional edge between an extra VD node and a VD node,
10. Directed edge from an extra VD node to a VD node in a bump pad.

Type-6 edges are constructed from an I/O pad to its surrounding extra VD nodes. Every Type-7 edge (e.g., the edge between extra VD nodes 3 and 5) is constructed without crossing the PA wires (VD edges). Type-8 edges are constructed between two adjacent extra VD nodes around the same VD node to cross the PA wires. Type-9 edges consist of two categories. In the first category, if only one terminal v of a VD edge is on the routed PA wires (e.g., VD node 7), a Type-9 edge is constructed between the other terminal and the closest one of the extra VD nodes surrounding v (e.g., VD node 6 and extra VD node 3). In the second category, the edges are constructed between a VD node and its surrounding extra VD nodes. Type-10 edges are connected to a VD node in a bump pad from its surrounding extra VD nodes. Each edge is also associated with a $(cost, capacity)$ ordered-pair.

The cost of the Type-8 edge is $\gamma \times d_b \times n_w$, where γ is a user-defined positive constant and n_w is the number of the crossed PA wires. In the second category of the Type-9 edges, the cost is $\gamma \times d_b \times n_r$. If an edge is like the one connecting extra VD node 2 and the thick edges, n_r equals the n_w between extra VD nodes 1 and 2. Otherwise, $n_r = 0$ like the one on the edge between extra VD node 1 and the connected VD node. That is because there is no wire crossing to route into nets i and j. The cost of any other edge is the edge length l_e. The capacity of the Type-7 edge is defined as follows:

$$Capacity(e) = \frac{n_{max} - n_w}{\nu}. \quad (2)$$

Recall that each Type-7 edge is along a VD edge. On the VD edge, n_w is the number of routed PA wires and n_{max} is the maximum capacity. $\nu = 3$ if the VD edge is a thick one. That is because we can route wires into the middle of the thick edges except at the two sides. Otherwise, $\nu = 2$. The capacity δ of the Type-8 edge is infinity. In Section 3.2.4, we will discuss how to change δ to improve the routing results. For the edge in the first category of the Type-9 edges, its capacity equals n_{max} of the deleted VD edge between one VD node and the other VD node surrounded by extra VD nodes (e.g. VD nodes 6 and 7). In the second category of the Type-9 edges, the capacity is given by Equation 2 where $\nu = 3$. The capacity of the remaining edges is 1. After applying MCMF, Figure 7(a) shows the routing result of the FA nets. Net 5 crosses net 1 without detouring bump pad 1 since we only set high costs on the Type-8 edges and the edges in the second category of the Type-9 edges to avoid wire crossings. The reason is that ripping up and rerouting some PA nets may further improve the routing results.

3.2.4 *Iterative Improvement*

After the FA routing, there may be some wire crossings. The reasons are that there is no solution without wire crossings or the detours of the FA nets are too long. Then we rip up and reroute the PA nets crossing the routed FA wires. In Figure 7(a), the total additional cost of net 5 to cross net 1 on G is γ. In Figure 8, we rip up and reroute net 1 if and only if the increase of the total wirelength is less than γ, or there is no other routing path of net 5. If we choose not to rip up and reroute net 1, we will set the capacity δ of the edge which makes net 1 crossed to be 0. By doing so, we can forbid net 5 to cross net 1 again. Finally, the iterative improvement is converged.

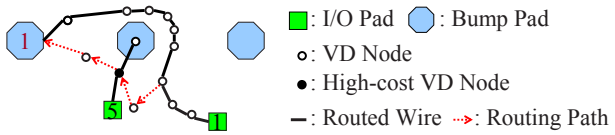

Figure 8: Improvement of Routing Results.

3.3 Detailed Routing

3.3.1 *Routing-Path Refinement*

After the global routing, only the edge with wires is left. In Figure 9(a), there are two FA nets and two PA nets. According to the number of wires (#Wires) on each edge, we can remove the VD nodes and refine the routing paths without generating any wire crossing as in [3]. Then, in Figure 9(b), we can split the edges of the VD nodes into independent wires with wire nodes.

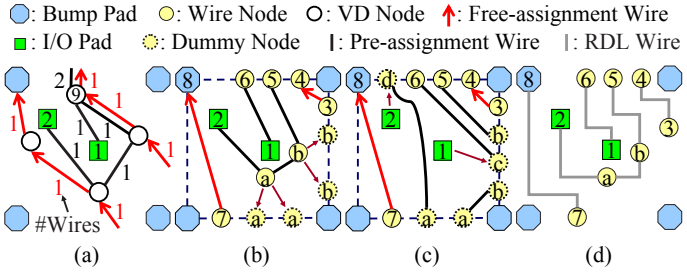

Figure 9: (a) Global-routing Paths. (b) Path Refinement. (c) Net-ordering Determination. (d) Maze Routing.

3.3.2 *Ordering Determination and Maze Routing*

In each tile, we can treat the routing as a channel routing. So we can modify the net-ordering determination algorithm presented in [3] to generate a routing sequence of the wires in a tile. Note that the net-ordering determination algorithm restricts each terminal to be on the boundary of a tile and to have no detours, but in the detailed routing problem, there may be I/O pads with detours inside a tile. Hence, in Figure 9(b) and (c), we insert dummy nodes instead of each I/O pad or wire node inside the tile. For example, since wire node a shares the same VD node with wire node 7, two dummy nodes a are inserted at the right side of wire node 7 to avoid crossing net 8. Since all terminals are on the boundary now, we can apply the determination algorithm. The resulting routing sequence $S =<$ $w(8,7), w(a,b), w(3,4), w(b,5), w(1,6), w(2,a) >$ and then we can route each wire $w(i,j)$ in a tile without intersecting each other by maze routing [3]. Figure 9(d) shows the result of maze routing.

4. 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, listed in Table 1, are real industry designs. In Table 1, "Circuits" is the names of circuits, "#I/O Pads (Free-/Pre-assignment)" is the number of I/O pads of FA/PA nets, and "#Bump Pads" is the number of bump pads.

Table 1: Benchmarks for the Package-Board Co-design.

Circuits	#I/O Pads (Free-/Pre-assignment)	#Bump Pads
fcpb1	162 (68/94)	289
fcpb2	301 (117/184)	400
fcpb3	346 (153/193)	441
fcpb4	333 (137/196)	441
fcpb5	1059 (394/665)	1764

Table 2: Comparison between FA-maze, PF-maze, and Ours. (N/A: Not Available.)

Routers / Circuits	Routability (%)			Total Wirelength (um)			CPU Time (s)		
	FA-maze	PF-maze	Ours	FA-maze	PF-maze	Ours	FA-maze	PF-maze	Ours
fcpb1	86.42	90.74	100	N/A	N/A	4539121	11	19	33
fcpb2	76.74	85.38	100	N/A	N/A	9920878	28	50	145
fcpb3	80.06	88.15	100	N/A	N/A	15431763	40	84	163
fcpb4	76.88	93.99	100	N/A	N/A	16584677	35	82	164
fcpb5	77.81	93.39	100	N/A	N/A	192887686	1124	1655	3438
Average	79.58	90.33	100						

We compared our algorithm with two heuristics, namely FA-maze and PF-maze. FA-maze integrates maze routing with some techniques presented in [3]. In FA-maze, it first tried to optimize the routing of FA nets by [3]. Then, the PA nets were routed sequentially by maze routing. PF-maze is our algorithm without performing the congestion estimation and the iterative improvement. Therefore, we set the values of α, β, and γ in PF-maze to be 0, 0, and ∞, respectively. In our proposed algorithm, in contrast, the values of α, β, and γ are set to be 3, 1, and 5, respectively. To fairly compare the three algorithms, we applied the same routing sequence of the PA nets. The routing sequence was decided by the non-decreasing order of the Manhattan lengths of the nets. The experimental results are shown in Table 2. We report the routability, the total wirelength, and the CPU times. Compared with FA-maze, our algorithm improves the routability by 20.42%. Note that for all circuits, FA-maze fails to find a routing solution while ours can achieve 100% routability. Compared with PF-maze, our algorithm improves the routability by 9.67%, which reveals the effects of the congestion estimation and the iterative improvement. The results show that our unified FA and PA routing algorithm is effective and efficient for the PB co-design.

5. CONCLUSIONS

We have developed a unified FA and PA area-I/O flip-chip router for PB co-design. Our DT- and VD-based network-flow algorithm guarantees to find the optimal solution with the minimum wirelength for the free-assignment nets. Our PB co-design routing flow can facilitate the interactions among free- and pre-assignment nets and lead to superior routing solutions with 100% routability and the shorter routed wirelength.

6. REFERENCES

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

[2] J.-W. Fang, C.-H. Hsu, and Y.-W. Chang, "An integer linear programming based routing algorithm for flip-chip design," *Proc. of DAC*, pp. 606–611, 2007.

[3] J.-W. Fang and Y.-W. Chang, "Area I/O flip-chip routing algorithm for chip-package co-design," *Proc. of ICCAD*, pp. 518–522, 2008.

[4] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.

[5] UMC, "0.13μm flip-chip layout guideline," p. 6, 2004.

[6] M.-F. Yu, J. Darnauer and W.-M. Dai, "Interchangeable pin rouing with application to package layout," *Proc. of ICCAD*, pp. 668–673, 1996.