Routing for Chip-Package-Board Co-Design Considering Differential Pairs

Jia-Wei Fang[1], *Kuan-Hsien Ho*[1], *and Yao-Wen Chang*[1,2]
[1]*Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan*
[2]*Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan*

***Abstract*— Nanometer effects have complicated the designs of chips as well as packages and printed circuit boards (PCB's). In order to improve the performance, convergence, and signal integrity of the design, chip-package-board co-design is strongly recommended by industry. In this paper, we present the first routing algorithm in the literature for chip-package-board co-design with differential-pair considerations. Our algorithm is based on linear programming and integer linear programming and guarantees to find an optimal solution for the addressed problem. It first creates global-routing paths among chips, packages, and a PCB. Without loss of the solution optimality, our routing formulation can reduce the numbers of integer variables (constraints) by 95% (99%) on average. Then, any-angle routing is applied to complete the routing. Experimental results based on five real industry designs show that our router can achieve 100% routability and the optimal global-routing wirelength and satisfy all differential-pair constraints, under reasonable CPU times, whereas recent related work results in much inferior solution quality.**

I. INTRODUCTION

A. Chip-Package-Board Co-Design

Due to the nanometer technology, modern VLSI designs contain greater functionality and require much higher performance than ever. Consequently, the I/O count increases significantly, and more I/O signals come out of a chip through a package. How to complete the routing for such a large number of I/O pins has become a big challenge to IC, package, and printed circuit board (PCB) designers. Figure 1 shows an example of a chip, a Ball Grid Array (BGA) package, and a PCB. The package routing is to route a net from a finger to a bump ball, while the PCB routing is to route a net from a bump ball to a pad.

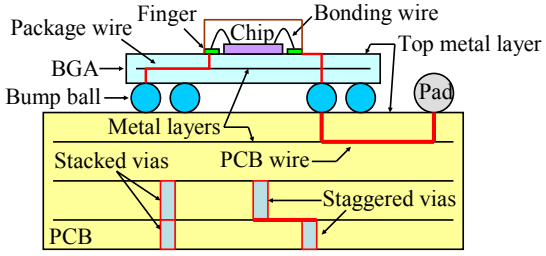

Fig. 1. Cross sections of a chip, a BGA package, and a PCB.

Due to the large I/O count of a chip, design performance would be substantially degraded without considering chip-package-board (CPB) co-design for I/O planning. The reason is that if a chip designer dose not use the information of package and PCB to deploy I/O's in the early design stage, it might result in unsolvable design violations in the later package and PCB designs. Furthermore, good I/O planning can have many advantages such as a smaller package size, shorter wirelength, higher routability, and fewer vias which also imply better reliability. Therefore, it is desirable to address the I/O planning problem of package generation, I/O pin assignment, and package and PCB routing for CPB co-design.

B. Differential Pairs

Differential-pair (DP) routing is a popular technique for modern high-speed PCB designs because differential pairs (DP's) have high noise immunity, electromagnetic interference reduction, and ground bounce insensitivity, which are all critical issues in high-speed signal transmission on a PCB (and even on a package or a chip). Figure 2 illustrates a DP transmission structure. Each DP consists of two complementary signals used to transmit one signal. The input signal is first encoded into a positive part and a negative counterpart. Then, these two signals are transmitted in close proximity along the routing channel. Since the signal wires are routed close to each other, the noises on the channel can be simultaneously absorbed by the two signals. As a result, the noises can be cancelled, and the original signal can be transformed back. In order to utilize the advantages of DP's, the two nets of a DP shall be routed parallel to each other with similar wirelength and far away from other nets. Therefore, it is desirable to consider DP's to improve the performance and signal integrity during CPB co-design.

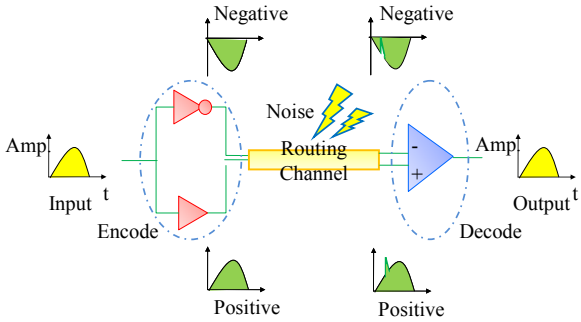

Fig. 2. Transmission structure of a differential pair.

C. Previous Work

To the best knowledge of the authors, there is no previous work in the literature to address the routing problem of CPB co-design considering DP's. Some related works [1], [2], and [5] are proposed for BGA routing, flip-chip routing, PCB routing, and package-board co-design. The work [1] proposed a package-pin reassignment algorithm based on simulated annealing (SA) to improve the routability considering package and PCB co-design. However, the algorithm cannot guarantee 100% routabiliy of package and PCB routing after pin reassignment. Furthermore, as shown in Figure 1, it also did not consider stacked vias [6] which can have better reliability than staggered ones. The work [5] proposed a greedy heuristic for global BGA routing which did not consider co-design and I/O planning. The work [2] applied integer linear programming (ILP) for single-layer flip-chip routing. It used integer variables and constraints to formulate the wire-crossing problem. However, the formulation is not efficient for CPB co-design. The details will be discussed in Section IV.

D. Our Contributions

In this paper, we present the *first* routing algorithm for CPB co-design. Our work also considers differential-pair routing, routability optimization under stacked vias, and total wirelength minimization. Our algorithm first constructs two independent routing networks for both DP's and other nets to find the routes among fingers, bump balls, and pads. Each routing network can be formulated as a *linear programming (LP)*. To achieve optimal solutions, we apply ILP to solve all the routing networks simultaneously. However, ILP is NP-complete [3] and thus computationally expensive. Consequently, we propose two methods to reduce the numbers of variables and constraints. Without loss of the solution optimality, our routing formulation can improve the numbers of integer variables (constraints) by 95% (99%) on average. Then, any-angle routing is applied to complete the routing. Experimental results based on five industry designs show that our router can achieve 100%

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

routability and the optimal global-routing wirelength, and satisfy all differential-pair constraints, under reasonable CPU times.

The rest of this paper is organized as follows. Section II gives the formulation of the routing problem of CPB co-design. Section III details our routing algorithm. Section IV shows the experimental results. Finally, conclusions are given in Section V.

II. Problem Formulation

Figure 3 shows the structure of the chip, package, and PCB routing; On the PCB, one component and one BGA package with a chip are placed. There are two types of nets for routing: one from fingers to bump balls, and one from fingers to pads via bump balls. Figure 3 illustrates the two types of routing. *Package routing* is referred to as a net routed from a finger to a bump ball. The net ended at a bump ball will be used to connect to the PCB (and other packages). *PCB routing* is referred to as a net that connects to a finger routed from a bump ball to a pad (e.g., s_7 in Figure 3). A 2-pin net is referred to a net routed from a finger to a bump ball or to a pad via a bump ball, while a multi-pin one is referred to a set of 2-pin nets that share the same pad. A DP in the PCB routing is a pair of nets (or pads), such as s_5 and s_6 in Figure 3, which are used to transmit the same signal and routed in parallel to each other with similar wirelength.

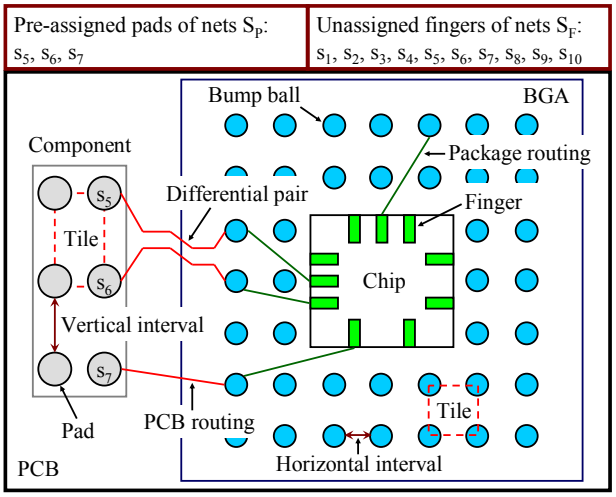

Fig. 3. A BGA package and a component on a PCB.

Let F be the set of fingers, B be the set of bump balls, and P be the set of pads. Let L_{BGA} and L_{PCB} denote the sets of BGA and PCB metal layers, respectively. We define a vertical/horizontal *interval* to be the segment formed by two adjacent pads/bump balls, and a *tile* to be the rectangle formed by four adjacent pads or bump balls. See Figure 3 for an illustration. Let S_P be the set of pre-assigned nets associated with pads, and S_D be the set of pre-assigned DP's associated with pads; therefore, $S_D \subseteq S_P$. All pads of nets are pre-assigned before routing, e.g., the pads with s_5, s_6, and s_7. Let S_F be the set of *unassigned* nets associated with fingers. Note that no net is pre-assigned to fingers before routing. Recall that the net ended at a bump ball will be used to connect to other devices. Therefore, $S_P \subseteq S_F$ since each pad assigned a net must be routed to a finger of the chip.

We formulate the addressed routing problem as follows:

Problem 1: Given a PCB with pads in a set of components, BGA packages with bump balls and chips with fingers, the numbers of BGA and PCB metal layers, a netlist (with specified differential pairs), and routing design rules, the routing problem for chip-package-board co-design considering differential pairs is to complete the package and PCB routing so that all nets are routed, no design rules (including differential-pair constraints) are violated, and the resulting total wirelength is minimized.

III. The Routing Algorithm

A. The Chip-Package-Board Co-Design Flow

Our CPB co-design flow, as shown in Figure 4, consists of two stages: PCB and package global routing followed by any-angle detailed routing.

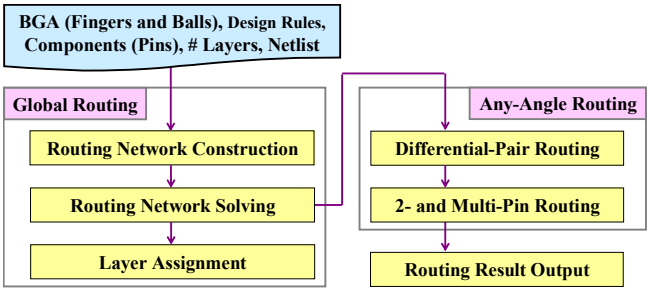

Fig. 4. The chip-package-board co-design flow.

In global routing, we first construct two routing networks G_N and G_D to formulate the routing of nets (could be 2- or multi-pin nets) and DP's by LP. To concurrently route 2- and multi-pin nets and DP's, we then merge the two routing networks G_N and G_D into a complete routing network G. The routing of the network G is formulated as an ILP due to the additional complexity incurred by wire congestions/crossings. Since ILP is NP-complete [3] and thus computationally expensive, we present a method to reduce the numbers of variables and constraints. Then, an ILP solver is used to solve the ILP and find the global routes of nets. After getting the global routes, layer assignment is applied to distribute nets to each metal layer with stacked vias [6] which can provide high reliability. The distributed global routes give the global-routing paths of the nets. Finally, any-angle detailed routing is applied to complete the routing. We detail the routing process in the following.

B. Global Routing

For global routing, we define six types of nodes as shown in Figure 5(a). The bump balls in the package and PCB routing are the same. In the PCB routing, for all tiles, we create a set T_{PCB} of PCB tile nodes. Each PCB tile node $t_{PCB}(i)$ represents a tile i. We also create a set M_{PCB} of PCB interval nodes. The PCB interval node $m_{PCB}(j)$ is introduced to model the interval j on an outer boundary of a component or a BGA (see the left of Figure 5(a)). For each BGA package, a set D_{PCB} of PCB nodes is created. Each PCB node $d_{PCB}(k)$ is inserted near the bump ball k. By using the PCB nodes, we can route nets from pads to bump balls. Similar to the PCB routing, we also create a set M_{BGA} of BGA interval nodes and a set T_{BGA} of BGA tile nodes for the package routing. A BGA interval node is introduced for each interval not on the outer boundaries of the BGA (see the right of Figure 5(b)). Around a chip, a set D_{BGA} of four BGA nodes are created. One BGA node $d_{BGA}(k)$ is created for the boundary k of the chip. Then we can use the BGA nodes to route nets from bump balls to fingers.

1) Basic LP Formulation of 2- and Multi-Pin Nets: Figure 5(b) shows the routing network G_N of 2- and multi-pin nets, and Figure 6 sketches the complete routing network G. We define a set Q of nodes and a set E of edges in G_N. By using the routing networks, we can route nets through the nodes and edges. There are 19 types of edges in E as follows:

- $e(source, p)$: directed edge from the source to a pad.
- $e(source, b)$: directed edge from the source to a bump ball.
- $e(f, sink)$: directed edge from a finger to the sink.
- $e(p, t_{PCB})$: directed edge from a pad to a PCB tile node.
- $e(p, d_{PCB})$: directed edge from a pad to a PCB node.
- $e(t_{PCB}(i), t_{PCB}(j))$/$e(t_{BGA}(i), t_{BGA}(j))$: bi-directional edge between two PCB (BGA) tile nodes.
- $e(t_{PCB}, m_{PCB})$/$e(t_{BGA}, m_{BGA})$: directed edge from a PCB (BGA) tile node to a PCB (BGA) interval node.
- $e(t_{PCB}, b)$: directed edge from a PCB tile node to a bump ball.
- $e(m_{PCB}, t_{PCB})$: directed edge from a PCB interval node to a PCB tile node.
- $e(m_{PCB}, d_{PCB})$: directed edge from a PCB interval node to a PCB node.
- $e(d_{PCB}, m_{PCB})$: directed edge from a PCB node to a PCB interval node.
- $e(d_{PCB}, b)$: directed edge from a PCB node to a bump ball.
- $e(d_{PCB}(i), d_{PCB}(j))$: bi-directional edge between two PCB nodes.
- $e(b, t_{BGA})$: directed edge from a bump ball to a BGA tile node.
- $e(b, d_{BGA})$: directed edge from a bump ball to a BGA node.

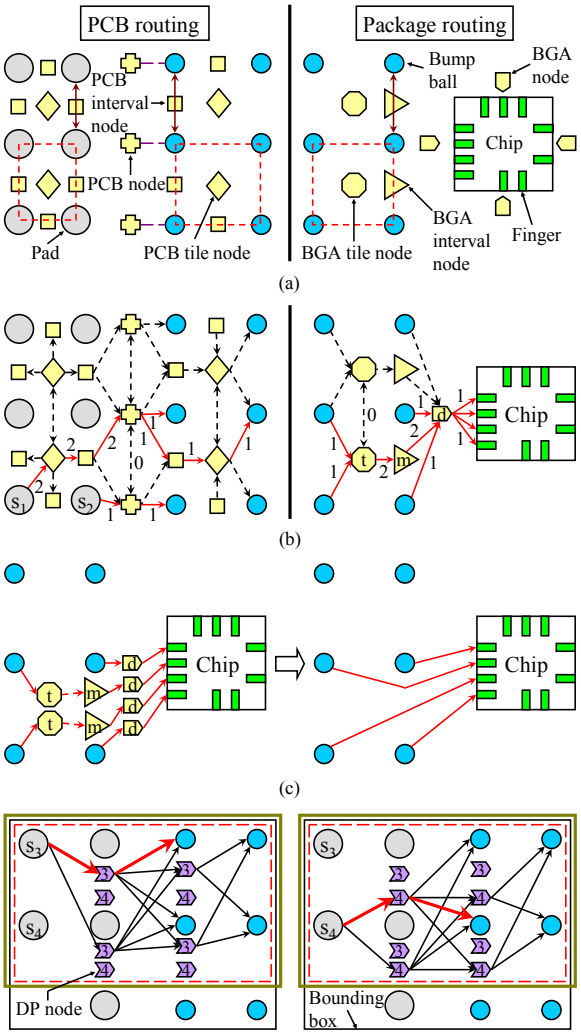

Fig. 5. (a) Nodes in the routing networks. (b) Routing network of 2- and multi-pin nets. (c) Refinement of routing paths. (d) Routing network of differential pairs.

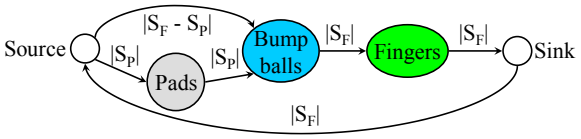

Fig. 6. Prototype of the complete routing network.

- $e(m_{BGA}, d_{BGA})$: directed edge from a BGA interval node to a BGA node.
- $e(d_{BGA}, f)$: directed edge from a BGA node to a finger.

$e(source, p)$, a pre-assigned net, is only constructed from the source to the pad p. $e(source, b)$ is constructed from the source to each bump ball b, and $e(f, sink)$ is constructed from each finger f to the sink. $e(p, t_{PCB})$ is constructed when the PCB tile node t_{PCB} is the closest one to the pad p. $e(p, d_{PCB})$ is constructed when the PCB node d_{PCB} is the closest one to the pad p. $e(t_{PCB}(i), t_{PCB}(j))$ and $e(t_{BGA}(i), t_{BGA}(j))$ are constructed when two PCB or BGA tile nodes are adjacent. $e(t_{PCB}, m_{PCB})$, $e(m_{PCB}, t_{PCB})$, and $e(t_{BGA}, m_{BGA})$ are constructed when a PCB (BGA) tile node and a PCB (BGA) interval node are in the same tile. $e(t_{PCB}, b)$ is constructed when the PCB tile node t_{PCB} and the bump ball b are in the same tile. $e(m_{PCB}, d_{PCB})$, $e(d_{PCB}, m_{PCB})$, and $e(m_{BGA}, d_{BGA})$ are constructed when a PCB (BGA) node is the closest one to a PCB (BGA) interval node. $e(d_{PCB}, b)$ is constructed when the bump ball b is the closest one to the PCB node d_{PCB}. $e(d_{PCB}(i), d_{PCB}(j))$ is constructed when two PCB nodes are adjacent. $e(b, d_{BGA})$ $(e(b, t_{BGA}))$ is constructed when a BGA (tile) node is the closest one to the bump ball b. $e(d_{BGA}, f)$ is constructed when the BGA node d_{BGA} and the finger f are on the same side of the chip.

According to the routing network in Figure 5(b), the PCB and BGA nodes are used to reduce the number of edges. For example, if we do not use BGA nodes, each bump ball and BGA interval node need to be connected to every finger, thus incurring a large number of edges.

In Figure 6, the size of a set associated with an edge gives the number of essential nets routed from one node to the other. $|S_F|$ nets need to be assigned to fingers. Since $|S_P|$ $(S_P \subseteq S_F)$ nets are assigned from pads, the other $|S_F - S_P|$ nets need to be assigned from bump balls. On the other hand, the $|S_F - S_P|$ nets are the nets from bump balls to fingers.

Now we can formulate the routing network of 2- and multi-pin nets as an LP. The notations used in the LP formulation are as follows:

- $l(e(i,j))$: the known length of edge $e(i,j)$.
- $f(e(i,j))$: real variable that denotes the number of routed wires of edge $e(i,j)$.
- $n(i)$: the known number of wires connected to the pad i (2- or multi-pin nets).
- $w_{BGA}(i)$ $(w_{PCB}(i))$: the known maximum number of wires passing through a BGA (PCB) tile/interval i.
- $w_{BGA}(i,j)$ $(w_{PCB}(i,j))$: the known maximum number of wires passing through two adjacent BGA (PCB) tile nodes i and j.

Consequently, the routing problem can be formulated as follows:

$$\min \sum_{e(i,j)\in E} l(e(i,j)) \times f(e(i,j))$$

$$\textit{subject to} \quad \sum_{e(i,j)\in E} f(e(i,j)) = n(i), \forall i \in P, \tag{1}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) = 1, \forall j \in F, \tag{2}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) \le 1, \forall i \in B, \tag{3}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) = \sum_{e(j,k)\in E} f(e(j,k)), \forall j \in Q, \tag{4}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) \le |L_{BGA}| \times w_{BGA}(i), \forall i \in M_{BGA} \cup T_{BGA}, \tag{5}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) \le |L_{PCB}| \times w_{PCB}(i), \forall i \in M_{PCB} \cup T_{PCB}, \tag{6}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) \le |L_{BGA}| \times w_{BGA}(i,j), \forall i,j \in T_{BGA}, \tag{7}$$

$$\sum_{e(i,j)\in E} f(e(i,j)) \le |L_{PCB}| \times w_{PCB}(i,j), \forall i,j \in T_{PCB}, \tag{8}$$

$$\forall f(e(i,j)) \ge 0$$

The objective function is to minimize the total wirelength. Constraints (1)–(2) ensure 100% routability by forcing all fingers and pads with pre-assigned nets to be routed to bump balls. Constraint (3) ensures that at most one net connects to a bump ball. Constraint (4) makes the number of wires routed into a node equal to that routed from the node. Constraints (5)–(8) avoid wire congestions in every interval and tile of all metal layers.

According to the integrality theorem in [7], applying integer data to our routing network can get optimal solutions with each variable being an integer. Therefore, after solving the LP formulation, we can get an optimal legal solution without any incomplete wire. The solid edges in Figure 5(b) show the routing result where s_1 is a multi-pin net. The number associated with an edge denotes the number of wires. Then we refine the routing paths as shown in Figure 5(c). The method of the refinement is that for each edge with a non-zero integer solution i, we

split the edge and its two terminals into i wires. For example, at the right routing network of Figure 5(b), since the solution of $e(t,m) = 2$, we split $e(t,m)$ into two dashed wires as shown in the left of Figure 5(c). Therefore, in the right of Figure 5(c), all nets are independent without any wire crossing. Note that no wire crossing implies fewer vias and better reliability in the package and PCB routing. Finally, each finger not connecting to a pad is assigned a net $s_i \in S_F - S_P$.

We have the following theorem:

Theorem 1: Given the routing network G_N of 2- and multi-pin nets, the LP formulation can find the minimum global-routing wirelength with no wire crossing.

2) Basic LP Formulation of Differential Pairs: For PCB design, DP's are an important concern for better signal integrity. Figure 5(d) shows the routing network G_D for the DP (s_3, s_4). The case of multiple DP's can be handled in the similar way. Since each net of a DP usually has a wirelength constraint, we can draw a bounding box for the net. For example, the dashed bounding box for the pad of s_3, and the solid one for the pad of s_4. The distance from a pad to any position inside its bounding box must meet its wirelength constraint. Therefore, we can find the solution only in the intersection of the two bounding boxes (the thick bounding box). Then we create a set D_P of DP nodes for differential-pair routing. For each interval, we create two DP nodes i and j for s_i and s_j, respectively. Each DP node i represents the node which may be passed through by s_i in an interval and is named by the net. As shown in Figure 5(d), we use the DP nodes in vertical intervals as an example. Those in horizontal intervals can be created in the same way. The two DP nodes 3 and 4 are inserted into every interval. Since the two pads/nets of a DP shall be routed in parallel to each other, the sequence of each pair of two adjacent DP nodes has to be decided according to the sequence of the two pads. For example, since the pad of s_4 is under the pad of s_3, the DP node 4 has to be also under the DP node 3 in every interval. Thus, the two pads of s_3 and s_4 are always routed in parallel to each other if they pass through the same intervals.

We define the set E_D of four types of edges as follows:

- $e(p, d_p)$: directed edge from a pad to a DP node.
- $e(p, b)$: directed edge from a pad to a bump ball.
- $e(d_p(i), d_p(j))$: directed edge from a DP node to another one.
- $e(d_p, b)$: directed edge from a DP node to a bump ball.

$e(p, b)$ is not shown in Figure 5(b), and $e(p, d_p)$ is constructed when a pad and a DP node are of the same net. $e(d_p(i), d_p(j))$ is constructed when the two DP nodes are of the same net. All edges are forbidden to cross through any vertical interval since we want to use an edge to exactly represent a net passing through a particular interval. As shown in Figure 5(d), the left routing network shows all the routing paths of s_3, and the right routing network shows all the routing paths of s_4. Then we can formulate the routing networks as an LP. Note that the construction of the package routing network of DP's is the same as the right of Figure 5(b).

The new notations used in the LP formulation are as follows:

- $\phi_k(e(i,j))$: if $e(i,j)$ is in the tile/interval k, $\phi_k(e(i,j)) = 1$; otherwise, $\phi_k(e(i,j)) = 0$.
- $\psi(i,j)$: if the two DP nodes (bump balls) i and j are adjacent in the same interval, $\psi(i,j) = 1$; otherwise, $\psi(i,j) = 0$.
- $\hat{I}_{BGA}/\hat{I}_{PCB}$: set of intervals in the BGA package (PCB).
- $\hat{T}_{BGA}/\hat{T}_{PCB}$: set of tiles in the BGA package (PCB).

Then the DP routing problem can be formulated as follows:

$$\min \sum_{e(i,j)\in E_D} l(e(i,j)) \times f(e(i,j))$$

$$subject\ to \quad \sum_{e(i,j)\in E_D} f(e(i,j)) = 1, \forall i \in P, \tag{9}$$

$$\sum_{e(i,j)\in E_D} f(e(i,j)) \le 1, \forall j \in B, \tag{10}$$

$$\sum_{e(i,j)\in E_D} f(e(i,j)) = \sum_{e(j,k)\in E_D} f(e(j,k)), \forall j \in D_P, \tag{11}$$

$$\sum_{e(i,g)\in E_D} \sum_{e(j,h)\in E_D} \psi(i,j) \times (f(e(i,g)) - f(e(j,h))) = 0, \tag{12}$$
$$\forall i,j \in D_P,$$

$$\sum_{e(i,g)\in E_D} \sum_{e(j,h)\in E_D} \psi(i,j) \times \psi(g,h) \times \tag{13}$$
$$(f(e(i,g)) - f(e(j,h))) = 0, \forall i,j \in D_P, \forall g,h \in B,$$

$$\sum_{e(i,j)\in E_D} \phi_k(e(i,j)) \times f(e(i,j)) \le 2 \times |L_{PCB}|, \tag{14}$$
$$\forall k \in \hat{I}_{PCB} \cup \hat{T}_{PCB}, \forall f(e(i,j)) \ge 0.$$

The objective function is also to minimize the total wirelength. Constraint (9) guarantees 100% routability by forcing all pads of DP's to be routed to bump balls. Constraint (10) ensures that at most one net connects to a bump ball. Constraint (11) makes a wire routed into a node also routed from the node. Constraint (12) makes the two nets of a DP routed together in each interval. Constraint (13) makes a DP routed to two adjacent bump balls. Constraint (14) ensures that only one DP (two nets) is routed in a tile/interval of a metal layer because a DP shall be routed far away from other nets for better signal integrity.

Note that since we route the two nets of a differential signal together in every tile/interval, the wirelengths of them should be the same after they meet at the first adjacent DP nodes. Besides, the objective function will force the two pads of a DP routed near the midline of them, thus resulting in a balanced wirelength. After the differential-pair routing, therefore, the wirelengths of the two nets of a DP are similar.

According to the integrality theorem in [7], we can get an optimal legal solution with each variable being an integer. The thick edges in Figure 5(d) show the routing result for PCB routing.

We thus have the following theorem:

Theorem 2: Given the routing network G_D of DP's, the LP formulation can route the two nets of a DP in parallel to each other.

3) The ILP Formulation of Concurrent Routing: To concurrently route 2- and multi-pin nets and DP's, we merge the two routing networks G_N and G_D (Figures 5(b) and (d)) into the complete routing network G. Note that applying LP to G may result in unsolvable wire crossings between DP's and other nets. For example, one DP net and one 2-pin net cross each other in a PCB tile which can be passed through by two wires. The reason is that in PCB routing, G_N and G_D are two independent routing networks. Furthermore, DP's shall be routed far away from other nets for better signal integrity. Therefore, we apply ILP to solve the wire crossings between DP's and other nets in every tile/interval in a PCB. The new notations used in the ILP formulation are as follows:

- $u(k)$: 0-1 integer variable; if any $f(e(i,j))$ $(e(i,j) \in E_D)$ is larger than or equal to one in a PCB interval/tile k, $u(k) = 1$; otherwise, $u(k) = 0$.
- $w(k)$: the known maximum number of wires passing through an interval/tile in each metal layer.
- N_L: a known large real number.

Then we solve the following ILP formulation.

$$\min \sum_{e(i,j)\in E\cup E_D} l(e(i,j)) \times f(e(i,j))$$

$$subject\ to \quad Constraints\ (1) - (14), \tag{15}$$

$$\sum_{e(i,j)\in E_D} \phi_k(e(i,j)) \times f(e(i,j)) \le N_L \times u(k), \tag{16}$$
$$\forall k \in \hat{I}_{PCB} \cup \hat{T}_{PCB},$$

$$\sum_{e(i,j)\in E_D} \phi_k(e(i,j)) \times f(e(i,j)) \ge u(k), \tag{17}$$
$$\forall k \in \hat{I}_{PCB} \cup \hat{T}_{PCB},$$

$$\sum_{e(i,j)\in E} \phi_k(e(i,j)) \times f(e(i,j)) \le (|L_{PCB}| - u(k)) \times w(k), \tag{18}$$
$$\forall k \in \hat{I}_{PCB} \cup \hat{T}_{PCB}.$$

The objective function is to minimize the total wirelength. Constraints (16)–(17) find every interval and tile passed by DP's in the PCB. Constraint (18) decreases the maximum number of nets of a tile/interal passed by DP's to avoid routing 2- and multi-pin nets through it in a certain PCB metal layer.

We apply two methods to reduce the numbers of integer variables and constraints in the ILP formulation. The first one is to use integer variables to formulate wire congestions of each tile/interval in the routing network G, and thus a tile/interval has at most one integer variable (see Constraints (16)–(17)). Different from the work in [2], we do not directly use integer variables to formulate wire crossings between edge $e \in E$ and edge $e_d \in E_D$. Therefore, the number of our integer variables is much smaller than that in the work [2]. comparative studies will be given in Section IV. The second method to prune the solution space is the bounding-box reduction as illustrated in Section III-B.2 (also see Figure 5(d)). Since there are wirelength constraints of DP's, we only need to set an integer variable $u(k)$ to represent the tile/interval k inside the thick bounding box.

After solving the ILP formulation and refining the routing paths of 2- and multi-pin nets in the PCB and all nets in the BGA package, we can get a routing result without any wire crossing in all metal layers. Therefore, we have the following theorem:

Theorem 3: Given the routing network G of 2- and multi-pin nets and DP's, the ILP formulation can find the minimum global-routing wirelength with no wire crossing in all metal layers.

4) Layer Assignment: In our routing network G, we integrate $|L_{BGA}| + |L_{PCB}|$ metal layers into one metal layer by increasing the routing resources of every interval/tile. Therefore, we need to do layer assignment for every net before any-angle detailed routing. We apply layer assignment to the PCB and the BGA routes. For PCB routes, we shall handle DP's first because assigning them to the upper metal layers implies fewer vias. Further, the two nets of a DP should be assigned to the same metal layer. As a result, we assign one of them first and simply assign the other net to the same layer. Then the nets of PCB routes can still be assigned into each metal layer because we consider the wire congestions between DP's and other nets in the routing network G. Figure 7 shows an example of the routing result on a complete routing network. We define a set R of wire segments of nets. A wire segment $r(i, j)$ represents the segment j of net i. Every net is divided into a set of wire segments by intervals. For example, in Figure 7(a), net 1 of the pad of s_1 is divided into three wire segments, $r(1, 1)$, $r(1, 2)$, and $r(1, 3)$. In this example, we assume that there are one PCB metal layer and two BGA metal layers. In the PCB routing (Figure 7(a)), since our ILP formulation considers the wire congestions between DP's such as (s_3, s_4) and other nets, we can route all nets successfully. In the package routing (Figure 7(b)), we assume that each tile/interval can only be passed through by one wire of every metal layer. Therefore, in the dashed tile, we need to apply layer assignment to nets i and g ($r(i, j)$ and $r(g, h)$). We formulate the layer assignment as an LP. The notations used in the LP formulation are as follows:

- $l_q(r(i, j))$: a real variable; $l_q(r(i, j)) = 1$ if $r(i, j)$ is assigned to metal layer q.
- $\phi_k(q, r(i, j))$: if $r(i, j)$ is in the interval/tile k of metal layer q, $\phi_k(q, r(i, j)) = 1$; otherwise, $\phi_k(q, r(i, j)) = 0$.
- $c(q)$: a user-defined cost of the metal layer q.

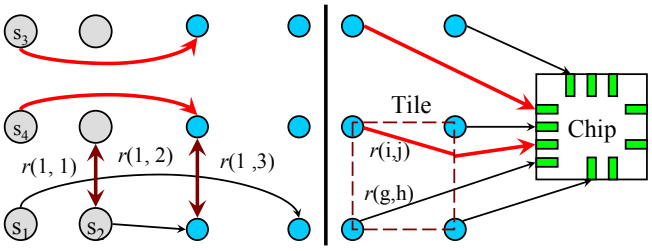

Fig. 7. (a) Routing result of the PCB. (b) Routing result of the BGA package.

Then we can give the LP formulation of the layer assignment as follows:

$$\min \sum_{q \in L_{BGA} \cup L_{PCB}} \sum_{r(i,j) \in R} c(q) \times l_q(r(i, j))$$

$$\textit{subject to} \sum_{q \in L_{BGA} \cup L_{PCB}} l_q(r(i, j)) = 1, \forall i \in S_F, \tag{19}$$

$$\sum_{q \in L_{BGA} \cup L_{PCB}} \sum_{r(i,j) \in R} \phi_k(q, r(i, j)) \times l_q(r(i, j)) \leq w(k), \tag{20}$$

$$\forall k \in \hat{T}_{PCB} \cup \hat{T}_{BGA} \cup \hat{I}_{PCB} \cup \hat{I}_{BGA},$$

$$l_q(r(i, j)) - l_q(r(i, k)) = 0, \forall j, k \in i, \forall q \in L_{BGA} \cup L_{PCB} \tag{21}$$

$$0 \leq l_q(r(i, j)) \leq 1, \forall l_q(r(i, j)).$$

The objective function is to minimize the number of total vias under the maximum number of stacked vias constraint. Constraint (19) forces each net to be assigned to only one BGA/PCB metal layer. Constraint (20) avoids the wire congestion of each tile/interval in every metal layer. Constraint (21) gets the maximum number of stacked vias by forcing each wire segment of a net to be assigned in the same BGA/PCB metal layer.

We set $c(q) = 0$ if j is not a finger/pad because the number of vias of a net in the BGA/PCB routing can only be counted once. In the BGA package, since fingers are in the top metal layer and bump balls are in the bottom metal layer, there is at least $|L_{BGA}| - 1$ vias of each net. Therefore, we set $c(q)$ to a constant. In the PCB, since pads and bump balls are in the same metal layer, the upper layer implies fewer vias. Thus, we set $c(q)$ to a smaller number in the upper metal layer.

The LP formulation satisfies the integrality theorem in [7]. Therefore, we can guarantee that each net is completely assigned to only one BGA/PCB metal layer, and the number of stacked vias of the net is also maximized. Furthermore, since we consider wire congestions in the routing network G, and there is no wire crossing among DP's or other nets, we can certainly find a feasible solution without wire crossing in each metal layer.

C. Any-Angle Routing

In order to efficiently perform the detailed routing, the fingers along the chip are sorted in counterclockwise order at the beginning. Then, our detailed router can process net by net according to the order. Since the constraints of DP nets are more stringent than other nets, two different routing strategies are proposed for DP routing and ordinary 2- and multi-pin net routing in the following.

1) Differential-Pair Routing: For DP routing, two signal nets should be routed in close proximity, and their wirelength difference cannot exceed a tolerance bound in order to smoothly couple with each other and hereby provide better electrical characteristics. For this purpose, we propose a DP routing strategy to transform a global routing result into a legal detailed DP routing result. The routing idea is illustrated in Figure 8. Figure 8(a) and Figure 8(b) show the global routing result and the corresponding detailed routing result of a DP based on our approach, respectively. The dotted ring (minimum spacing ring) represents the minimum spacing between a wire and a pad/bump ball.

The two nets of a DP can be routed in close proximity and satisfy the length constraint simultaneously if they are always parallel to each other within a desired distance. In fact, the parallel routing structure can be viewed as to connect a series of parallelograms (see Figure 8(b)). Each turn for two parallel nets is just like to process a parallelogram and then create a new parallelogram. Here, two DP nets route from "A" to "B" by the parallel topology with four parallelograms. The length of the two nets within the four parallelograms must be the same since the routing topology is just like to connect four different parallelograms. In the first step of our approach, therefore, the routing paths starting from the DP bump balls and from the DP pads can be individually routed toward their bisectors with the same space. If the length difference of the initial non-parallel net segments is small, the total length difference must also be small. By this observation, the length difference can be minimized by routing the DP nets toward their individual bisectors. Then, the two DP nets can be treated as a virtual single net, and the 2-pin net routing strategy can be applied to finish the routing. Finally, the detailed DP routing result can be obtained.

2) 2- and Multi-Pin Net Routing: Since the PCB routing does not allow any routing path with an acute angle, the router first checks every turning point to avoid an acute angle for each ordinary net. Once the router detects an acute corner, the two adjacent net segments can be cut off to generate two obtuse angles, as shown in Figure 8(c). Then, the spacing between the net to other components is estimated to avoid other design-rule violations. If the net is too close to other components and violates the design rule, a turning point can be added to change the routing topology. As illustrated in Figure 8(c), adding a turning point can effectively solve the minimum spacing violation.

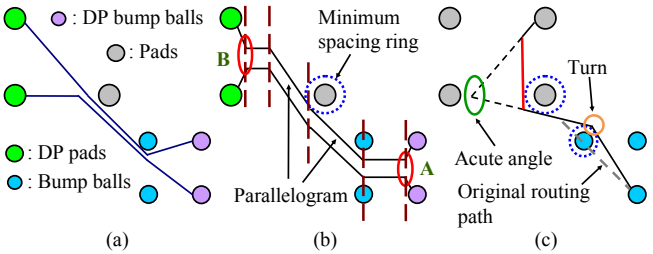

Fig. 8. (a) Global routing result of a differential pair. (b) Detailed routing result of a differential pair. (c) Correction of an acute-angle route.

IV. EXPERIMENTAL RESULTS

Our algorithm was implemented in the C++ programming language on a 2.8 GHz AMD Opteron Linux workstation with 8 GB memory. The public lp_solve [4] was applied to solve the LP and ILP. The benchmark circuits, listed in Table I, are real industry designs. In Table I, "Circuits" gives the names of circuits, "#Pads (DP/BGA)" lists the number of pads that are DP pads or connect to BGA, "#Fingers" gives the number of fingers, "#Balls" gives the number of bump balls, "#Nets (multi-/2-pin)" gives the number of multi-pin nets over all nets, and "#Metal layers (BGA/PCB)" gives the number of metal layers in the BGA and PCB, respectively.

TABLE I

BENCHMARK CIRCUITS FOR CHIP-PACKAGE-BOARD CO-DESIGN.

Circuits	#Compo-nents	#Pads (DP/BGA)	#Fingers	#Balls	#Nets (multi-/2-pin)	#Metal layers (BGA/PCB)
CPB1	2	74 (6/59)	150	159	3/150	1/1
CPB2	1	126 (8/58)	260	275	5/260	2/1
CPB3	3	192 (16/93)	429	453	12/429	2/1
CPB4	3	380 (28/274)	720	740	31/720	2/2
CPB5	4	683 (36/460)	1024	1075	51/1024	4/4

TABLE II

COMPARISONS AMONG SAR, WG, AND OURS.

Circuits	Routability (%)			#DP violations			Wirelength (mm)		
	SAR	WG	Ours	SAR	WG	Ours	SAR	WG	Ours
CPB1	76	100	100	1/3	0/3	0/3	643	1005	997
CPB2	84	100	100	2/4	0/4	0/4	1614	2033	2040
CPB3	79	100	100	5/8	0/8	0/8	4924	6789	6780
CPB4	87	100	100	10/14	0/14	0/14	10028	12263	12248
CPB5	80	N/A	100	11/18	N/A	0/18	21580	N/A	29402
Comp.	81	N/A	100	29/47	N/A	0/47			

Circuits	#Integer variables/constraints		CPU times (s)		
	WG	Ours	SAR	WG	Ours
CPB1	670/8430	45/135	2	510	6
CPB2	426/4938	35/105	5	456	16
CPB3	1106/13288	83/249	9	9737	34
CPB4	6010/69892	302/906	21	75641	195
CPB5	7968/100831	396/1188	96	$3*10^5$	366
Comp.	100%/100%	5%/1%			

Since there is no previous work on the addressed problem, one heuristic and one ILP based algorithms were implemented for the comparative study. The first algorithm SAR performs sequential assignment and routing from the top to the bottom metal layer based on maze routing. It routes DP's and then other nets. While routing a DP, it first routes one of the two DP pads to an un-assigned bump ball near the midline between them. The other one is then routed to other un-assigned bump balls following the path of the routed pad. Each pad of a 2- or a multi-pin net is routed to the closest un-assigned bump ball. If a bump ball is connected to a pad, it is routed to one of the fingers. Finally, all un-connected fingers are also routed to the closest un-assigned bump balls. The second algorithm WG is similar to ours. The only difference is that it applies ILP to formulate the wire-crossing constraints as in [2], instead of Constraint (18) as ours. Each edge e of G_D represents a $0-1$ integer variable $u(e)$. When an edge a of G_D and an edge b of G_N cross each other, the constraint $f(b) \leq (|L_{PCB}| - u(a)) \times w(k), \forall k \in \hat{I}_{PCB} \cup \hat{T}_{PCB}$ is added. It means that only one of a and b can be routed in each metal layer.

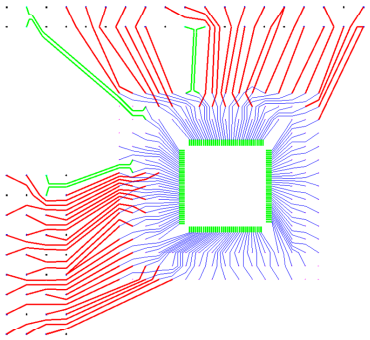

Fig. 9. CPB routing result for CPB1.

The experimental results are shown in Table II. Table II gives the routability, the total wirelength, the number of DP violations, the number of integer variables/constraints, and the CPU times. In the column of "#DP violations", the numerator gives the number of violations and the denominator gives the total number of DP's. As shown in the table, SAR results in the worst quality mainly because no concurrent assignment and routing is involved. Compared with WG, our algorithm can reduce the number of variables (constraints) by 95% (99%) on average. Further, without using our ILP formulation to solve the congestion between DP's and other nets incurs very long CPU times (even $>$ 3 days for a circuit), and is thus not feasible for this problem. The experimental results show that our algorithm can achieve 100% routability and the optimal global-routing wirelength and satisfy all differential-pair constraints in reasonable CPU times, due to the concurrent assignment and routing and much smaller numbers of integer variables/constraints. Figure 9 shows the routing result of CPB1. The red, green, blue wires are PCB, DP, and BGA wires, respectively.

V. CONCLUSIONS

We have developed the first router for CPB co-design in the literature, considering DP's and total wirelength minimization. Our ILP-based algorithm guarantees to find an optimal solution for the addressed problem with concurrent assignment and routing and much fewer integer variables/constraints. Experimental results have demonstrated that our routing algorithm is very effective and efficient for CPB co-design.

ACKNOWLEDGMENTS

This work was partially supported by Etron, SpringSoft, TSMC, and National Science Council of Taiwan under Grant No's NSC 96-2628-E-002-248-MY3, NSC 96-2628-E-002-249-MY3, NSC 96-2221-E-002-245, NSC 96-2752-E-002-008-PAE, and NSC 096-2917-I-002-120. The authors would like to thank Professor Ting-Chi Wang of National Tsing Hua University, Taiwan for his very constructive comments.

REFERENCES

[1] S.-S. Chen, W.-D. Tseng, J.-T. Yan, and S.-J. Chen, "Printed circuit board routing and package layout codesign," *Proc. of APCCAS*, pp. 155–158, 2002.

[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] M. R. Garey and D. S. Johnson, *A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[4] http://lpsolve.sourceforge.net/5.5/.

[5] Y. Kubo and A. Takahashi, "A global routing method for 2-layer ball grid array packages," *Proc. of ISPD*, pp. 36–43, 2005.

[6] F. H. Liu, G. E. White, V. Sundaram, A. O. Aggarwal, S. M. Hosseini, D. Sutter, and R. R. Tummala, "A novel technology for stacking microvias on printed wiring board," *Proc. of ECTC*, pp. 1134–1139, 2003.

[7] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, Springer, 2007.