Escape Routing for Staggered-Pin-Array PCBs

Yuan-Kai Ho, Hsu-Chieh Lee, and Yao-Wen Chang, *Fellow, IEEE*

***Abstract*—To accommodate the ever-growing pin number of complex printed circuit board (PCB) designs, the staggered pin array is introduced for modern designs with higher pin density. However, the escape routing for staggered pin arrays, which is a key component of PCB routing, is significantly different from that for grid arrays. This paper presents a routing algorithm for the escape routing for staggered-pin-array PCBs. We first analyze the properties of staggered pin arrays, and propose an orthogonal-side wiring style that fully utilizes the routing resource of the staggered pin array. A linear programming/integer linear programming-based algorithm is presented to solve the staggered-pin-array escape routing problem. Experimental results show that our approach successfully routes all test cases efficiently and effectively.**

***Index Terms*—Linear programming, physical design, printed circuit board, routing.**

I. Introduction

Escape ROUTING is a key problem in printed circuit board (PCB) design. The objective of this problem is to route nets from pins to the boundary of a component. According to the demanded ordering of the escaped pins in the component boundary, the escape routing problem can be classified into three types [2]: unordered escape routing [3]–[11], ordered escape routing [12], [13], and simultaneous escape routing [14]–[16]. Unordered escape routing routes nets without any ordering constraint along the boundary of a component, while ordered escape routing needs to conform to an ordering constraint along the boundary of a component. Simultaneous escape routing considers two components together; the ordering of the escaped nets along one component boundary needs to exactly match that of the escaped nets along the other component boundary. Each type of the escape routing problem has its practical applications. In this paper, the unordered escape routing problem is considered.

Manuscript received June 25, 2012; revised September 17, 2012; accepted March 25, 2013. Date of current version August 16, 2013. A preliminary version of this paper was presented at the 2011 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), San Jose, CA, USA, November 2011 [1]. This work was supported in part by IBM, SpringSoft, TSMC, Academia Sinica, and NSC of Taiwan under Grant NSC 101-2221-E-002-191-MY3, Grant NSC100-2221-E-002-088-MY3, Grant NSC 99-2221-E-002-207-MY3, and Grant NSC 99-2221-E-002-210-MY3. This paper was recommended by Associate Editor Dr. C. C.-N. Chu.

Y.-K. Ho and H.-C. Lee are with the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei 106, Taiwan (e-mail: yuankai@eda.ee.ntu.edu.tw; pg30123@eda.ee.ntu.edu.tw).

Y.-W. Chang is with the Graduate Institute of Electronics Engineering and Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, and also with the Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan (e-mail: ywchang@cc.ee.ntu.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2013.2259539

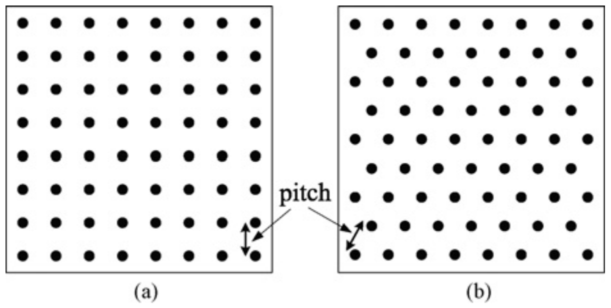


Fig. 1. Layout of two kinds of pin arrays. (a) Grid-pin array. (b) Staggered pin array.

Traditionally, the pin array is placed to form a grid-pin array, as shown in Fig. 1(a). However, as the feature size decreases, modern PCB designs become more and more complex. One of the critical technology issues is to satisfy the ever-increasing pin number in a fixed area. To increase the pin number, staggered pin arrays, which are formed by shifting specific columns of grid-pin arrays, are adopted to increase the pin number [17], [18]. Many industrial products adopt the staggered type, such as Intel processors on Socket 5 and Socket 7 [19]. Fig. 1(b) shows a staggered pin array. Compared with the grid-pin array, the staggered pin array offers higher pin density, which can be viewed from two perspectives as follows [20]: 1) the staggered pin array can accommodate more pins under the same pitch constraint and same area usage, and 2) the staggered pin array can get a larger pitch under the similar number of pins and same area constraints. One can easily verify that the staggered pin array in Fig. 1(b) has a higher pin number than the grid-pin array of the same area, shown in Fig. 1(a).

For the unorder escape routing problem, network flow is a popular method to solve the problem [3]–[10]. However, most of these network-flow-based methods focus on grid-pin arrays, but not on the staggered ones. To apply the network flow formulation, a square-tile model was proposed in [3]. However, the square-tile model cannot capture the valid routing for some specific cases. As a result, Yan *et al.* [4] proposed a correct network flow model to formulate square tiles. Since their model can only formulate square tiles, their model cannot be applied to staggered pin arrays. Fang *et al.* [6] used Delaunay triangulation and Voronoi diagrams to create triangular-tile models and solve the problem. Although this approach can be applied to staggered pin arrays, the angle of the triangular tiles of their model can be of any angle; as a result, their model might over restrict the capacity of each tile and could lose important solution space. For staggered pin arrays, Shi and Cheng [20] proposed some properties and only provided simple routing guidelines. Note that traditional grid-pin-array escape routing algorithms cannot be directly applied to the staggered pin arrays without losing significant solution space, since the horizontal and vertical wiring styles do not fit the staggered structure well. This issue will further be addressed in Section II.

In this paper, we propose an effective escape routing algorithm for staggered pin arrays. We first analyze the properties of staggered pin arrays, and propose an orthogonal-side wiring routing style that is specifically designed for the triangular-tile models in staggered pin arrays. Then we categorize the design rules into two types, while one of them needs additional constraints and more sophisticated handling. For these two types of design rules, we propose a linear programming (LP) formulation for the simpler type and an integer linear programming (ILP) formulation for the more sophisticated type. That is, given a staggered pin array with a set of design rules, we first determine whether the ILP formulation is needed for this set of design rules. Then either LP or ILP formulation is used to model this problem. In addition, we also propose a reduction technique to reduce the numbers of variables and constraints for our LP/ILP formulation. Experimental results show that our routing algorithm is effective and sufficiently efficient, and is thus suitable for staggered pin arrays.

We summarize our contributions as follows.

1) This paper presents the first work to analyze the routing in staggered pin arrays and derives a triangular-tile model that can fully utilize the routing resource.
2) Based on the triangular-tile model, a network-flow-based LP/ILP formulation is proposed to solve the global routing problem.
3) Experimental results show that our approach achieves 100% routability for all staggered-pin-array test cases in reasonable running time.

The rest of this paper is organized as follows. Section II analyzes the properties of the staggered pin arrays and points out the difficulty of the escape routing of staggered pin arrays. Section III gives the formulation of the routing problem. Section IV presents our escape routing algorithm for staggered pin arrays. Section V reports the experimental results. Finally, our conclusion is given in Section VI.

II. Preliminaries

A. Staggered Pin Array

An $m \times n$ staggered pin array is defined as follows: the pin array has m rows. There are n pins in each odd row, while there are $n - 1$ pins in each even row. The angle between the lines joining any adjacent two pins is always multiple of 60° [20]. Fig. 1(b) illustrates an example of a staggered pin array.

The staggered pin array accommodates more pins than the traditional grid-pin array under same pitch and area constraints. Take Fig. 1 as an example. The grid-pin array shown in Fig. 1(a) only accommodates 64 pins, while the staggered pin array shown in Fig. 1(b) can accommodate 68 pins. The increased ratio of the pin number in this case is 6.25%. Note that the two arrays have the same pitch and area. Besides, according to the analysis from Shi and Cheng [20], a larger staggered pin array can accommodate more pins than a grid-pin array with the same pith and area. For example, a 40×40 grid-pin array can accommodate 1600 pins, while the corresponding staggered one can accommodate 1817 pins. The increased ratio of the pin number is 13.56%.

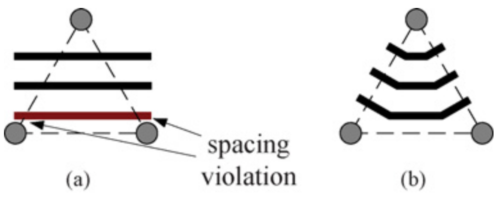


Fig. 2. Advantage of the orthogonal-side wiring style. (a) Only two horizontal wires can pass through the tile without violating DRC rules. (b) Three wires can pass through the tile by adopting the orthogonal-side wiring style.

B. Routing Style

Routing in a staggered pin array is more difficult (than routing in a grid one) because of its nonsquare structure. If we still adopt the vertical and horizontal routing style, the number of wires passing through the space between adjacent pins cannot achieve the maximum number. To utilize the space between adjacent pins efficiently, the wires should be orthogonal to the lines joining adjacent pins. We denote such wiring style as orthogonal-side wiring for convenience. Take the triangular tile in Fig. 2 for illustration. The maximum number of wires that can pass through adjacent pins is three according to the design rule checking (DRC) rules. However, if the horizontal wiring style is used, only two wires can pass the triangular tile, as shown in Fig. 2(a). On the other hand, if we use orthogonal-side wiring for routing, the number of wires that can pass through the tile is three, as shown in Fig. 2(b). That is to say, the space between adjacent pins can only be fully utilized for the orthogonal-side wiring style. If we directly apply the vertical and horizontal routing styles originally developed for the traditional square pin array to the staggered one, the routing space cannot be fully utilized, and a considerable part of solution space would be lost. This under-estimation of routing capacity between adjacent pins would increase wire length and/or decrease routability, thus greatly reducing the quality of routing results.

While routing by orthogonal-side wiring can fully utilize the routing space inside a triangular tile, we observe an issue when combining two adjacent tiles. Two triangular tiles are shown in Fig. 3, and each tile is passed by five wires. If we view each tile separately, there are no spacing constraint violations between pins and wires. However, the two innermost wires violate wire spacing rule if the following inequality holds:

$$\left\lfloor \frac{2(l-d)-2s}{w+s} \right\rfloor > \left\lfloor \frac{\sqrt{3}l-d-s}{w+s} \right\rfloor \tag{1}$$

where l is the pitch, d is the diameter of the pin, w is the wire width, and s is the wire spacing.

Since inequality (1) holds for the example in Fig. 3, the two innermost wires cannot be routed simultaneously. That

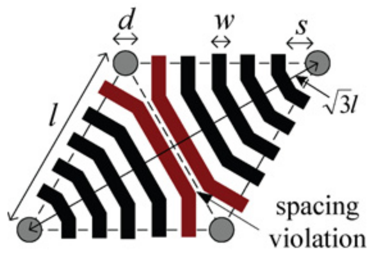


Fig. 3. Spacing between the two innermost wires violates design rules.

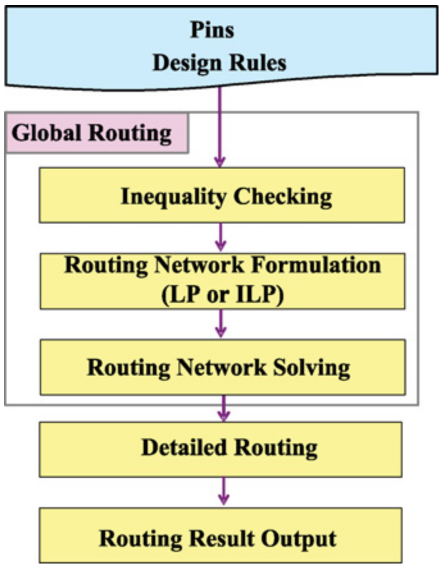


Fig. 4. Our proposed algorithm flow.

is, although five wires can pass through a single tile without violation, the maximum number of wires passing through these two adjacent tiles is nine, instead of 10.

III. Problem Formulation

The escape routing problem formulation here is similar to the traditional unorder escape routing problem [2], [4]. The key difference is that the pin array in our problem is in a staggered form. The problem can be defined as follows.

Escape Routing for Staggered Pin Arrays: Given an $m \times n$ staggered pin array with some marked pins to be routed to the boundary, and DRC rules including wire width, wire spacing, pin diameter, and pitch, the objective is to escape all marked pins to the array boundary so that no design rule is violated, and the total wire length is minimized under 100% routability guarantee.

IV. Routing Algorithm

In this section, we present our escape routing algorithm for staggered pin arrays. Our algorithm flow is shown in Fig. 4. We first check whether the input pin array satisfies inequality (1). Then, the routing network $G = (V, E)$ is constructed to represent the staggered pin array. If inequality (1) does not hold, we can use LP formulation on the routing network to solve the problem. On the other hand, if inequality (1) holds, we must formulate the routing with additional constraints to avoid wire spacing violation, and the formulation becomes an ILP problem. After the LP/ILP formulation is completed, we use the CPLEX [24] library to solve the formulation and acquire the global routing result. Finally, detailed routing is performed according to the global routing result, and it produces the routing result.

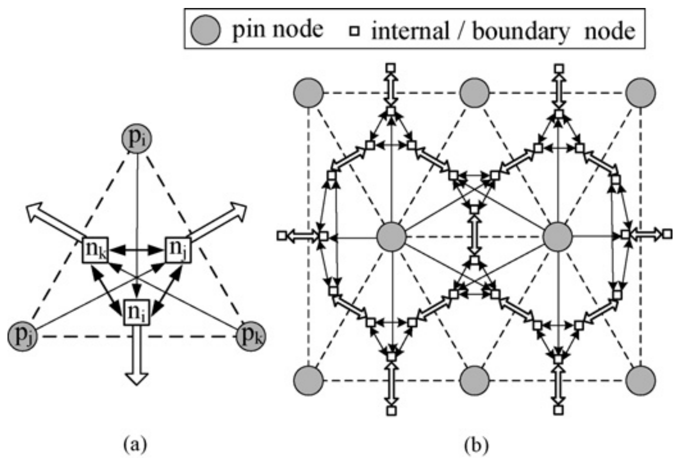


Fig. 5. (a) Proposed triangular-tile model. (b) Routing network constructed by combining all triangular-tile models in the pin array.

A. Routing Network Construction

To describe the construction of a global routing network, we start by explaining the construction of a triangular-tile model. A triangular-tile model represents a triangular region in the staggered pin array that is formed by three adjacent pins. After triangular-tile models are constructed, they can be connected to each other to form the global routing network G.

As shown in Fig. 5(a), three nodes, n_i, n_j, and n_k, are constructed for each triangular-tile model. These three nodes are connected to each other with undirected edges, and each of them is connected to a surrounding pin node with a directed edge. In addition, each internal node is also connected to either a corresponding internal node of adjacent tiles or a boundary node. Fig. 5(b) shows an example of the routing network which is formed by connecting each triangular-tile model.

We combine all triangular models together and create two extra nodes, source and sink, to form our routing network G. The source node is connected to all escape pins, and the sink node is connected to all boundary nodes. Thus, for the routing network G, the nodes in V can be categorized as follows.

1) $P_E = \{p_{e1}, p_{e2}, \ldots, p_{e|P_E|}\}$ is the set of escaped pins.
2) $P_U = \{p_{u1}, p_{u2}, \ldots, p_{u|P_U|}\}$ is the set of nonescaped pins.
3) $P = P_E \cup P_U$ is the set of all pins.
4) $N = \{n_1, n_2, \ldots, n_{|N|}\}$ is the set of internal nodes and boundary nodes.

The edges in E can be categorized as:

1) e_{source,p_i} is the directed edge form the source to a pin p_i;
2) $e_{n_i,\text{sink}}$ is the directed edge from an internal node n_i to the sink;

3) e_{p_i,n_j} is the directed edge from a pin p_i to an internal node n_j;
4) e_{n_i,n_j} is the directed edge from an internal node n_i to an internal node n_j.

B. LP Formulation

After the routing network is constructed, we can formulate the network flow problem with LP formulation. Some notations used in the LP formulation are:

1) $l(e_{i,j})$ denotes the length of edge $e_{i,j}$;
2) $f(e_{i,j})$ denotes the flow of edge $e_{i,j}$;
3) $c(e_{i,j})$ denotes the capacity of edge $e_{i,j}$. If $i \in P$, $c(e_{i,j}) = 1$. If $i, j \in N$, and i, j are not in the same tile, $c(e_{i,j})$ is assigned to be the maximum number of wires that can pass through the side crossing $e_{i,j}$ according to design rules; for other edges, $c(e_{i,j})$ is set to be infinite.

When inequality (1) does not hold, the wire spacing violation we discussed in Section II-B will not happen. Therefore, the problem can be solved with the following LP formulation:

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

subject to

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

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = |P_E|, \forall i \text{ is the source}, \tag{3}$$

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = |P_E|, \forall j \text{ is the sink}, \tag{4}$$

$$\sum_{e_{i,j} \in E} f(e_{i,j}) = \sum_{e_{j,k} \in E} f(e_{j,k}), \forall j \in P \cup N, \tag{5}$$

$$f(e_{i,j}) \le c(e_{i,j}), \forall e_{i,j} \in E, \tag{6}$$

$$\forall f(e_{i,j}) \ge 0. \tag{7}$$

The objective of the LP formulation is to minimize the total wire length. Constraints (2)–(4) ensure 100% routability by forcing all escaped pins to be routed to the sink (through boundary nodes). Constraint (5) is the flow conservation constraint that ensures the total flow income of a node equals the total output flow from the node. Constraint (6) ensures that the flow of an edge cannot exceed its capacity, thus avoiding design rule violations. Constraint (7) makes the flow on all edges nonnegative. Although our LP formulation seems to have real-number solutions, the constraints actually satisfy the unimodularity property [21], [22] because it is equivalent to the min-cost flow problem, and its optimal solution is guaranteed to be an integer solution. Ensuring the optimal solution being an integer solution, we avoid fractional wires on edges. Thus, detailed routing can be performed successfully. Based on the aforementioned discussions and the work [23], we have the following theorem.

Theorem 1: If inequality (1) does not hold, the escape routing problem for staggered pin arrays can be solved in polynomial time by our LP formulation on the routing network with $O(|P|)$ variables and $O(|P|)$ constraints, where P is the set of all pins (nodes).

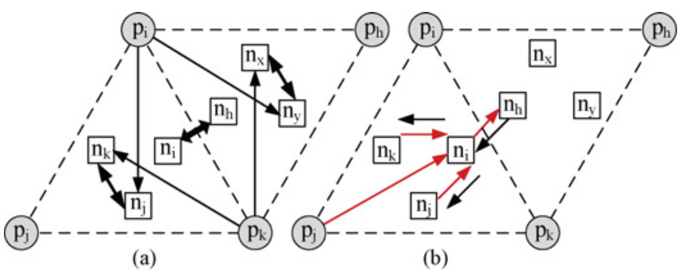


Fig. 6. Extra constraints for avoiding spacing violation.

Proof: Since inequality (1) does not hold, each triangular tile can accommodate maximum wires without wire spacing violation between adjacent tiles, as we discussed in Section II-B. We only need to formulate that the number of flows is no larger than its capacity in our constraints. Besides, the LP formulation satisfies the unimodularity property [21]. The optimal solution is guaranteed to be an integer solution. Thus, we can use our LP formulation to solve the escape routing problem in polynomial time [23] to get the minimum wire length and legal flows.

The number of LP variables is equal to the number of $f(e_{i,j})$. Obviously, this number is $O(|E|)$. However, the number of edges in a triangular model is constant, and the number of triangular models is linear to the number of pins. Thus, the number of ILP variables can be bounded by $O(|P|)$. The number of constraints (2) and (5) are both $O(|V|)$ and are linear to the number of all pins; the number of constraints (3) and (4) are both $O(1)$; the number of constraints (6) and (7) are both $O(|E|)$ and are linear to the number of pins according to the previous proof of the number of LP variables. Therefore, the total number of LP constraints is $O(|P|)$. ■

C. ILP Formulation

When inequality (1) holds, some additional constraints must be added to the LP formulation to avoid wire spacing violation, as discussed in Section II-B. That is, the number of wires that simultaneously pass through two adjacent tiles must be bounded with a certain number derived from design rules. Take the tiles in Fig. 3 for example, the two tiles in the figure can only accommodate nine wires, which should be confined with some additional constraints. We explain the constraints using Fig. 6 as an example. The number of wires that can pass through the two tiles in Fig. 6(a) is bounded by the distance between pins p_j and p_h. We define $\psi(p_j, p_h)$ to be the maximum wire number between pins p_j and p_h derived from design rules. We first analyze our tile model and identify the edges corresponding to wires crossing the line segment $\overline{p_j p_h}$; these edges are drawn in solid edges in Fig. 6(a). The set U_{p_j,p_h} is defined as the set of these edges, which includes e_{n_j,n_k}, e_{n_k,n_j}, e_{n_x,n_y}, e_{n_y,n_x}, e_{n_i,n_h}, e_{n_h,n_i}, e_{p_i,n_j}, e_{p_i,n_y}, e_{p_k,n_k}, and e_{p_k,n_x}. To avoid the wire spacing violation, we have the following constraint:

$$\sum_{e_{i,k} \in U_{p_j,p_h}} f(e_{i,k}) \le \psi(p_j, p_h). \tag{8}$$

The reason behind constraint (8) is that all flows of edges in U_{p_j,p_h} pass through the space between pins p_j and p_h, and their total flow must not exceed $\psi(p_j, p_h)$. In other words, the edges in U_{p_j,p_h} directly consume the capacity between pins p_j and p_h. Obviously, the edges e_{n_j,n_k}, e_{n_k,n_j}, e_{n_x,n_y}, e_{n_y,n_x}, e_{p_i,n_j}, e_{p_i,n_y}, e_{p_k,n_k}, and e_{p_k,n_x} consume the capacity. In addition, the edges e_{n_i,n_h} and e_{n_h,n_i} are parts of the following paths: $n_j \leftrightarrow n_i \leftrightarrow n_h \leftrightarrow n_x$, $n_k \leftrightarrow n_i \leftrightarrow n_h \leftrightarrow n_x$, $n_k \leftrightarrow n_i \leftrightarrow n_h \leftrightarrow n_y$, $n_j \leftrightarrow n_i \leftrightarrow n_h \leftrightarrow n_y$, $p_j \rightarrow n_i \rightarrow n_h \rightarrow n_x$, $p_j \rightarrow n_i \rightarrow n_h \rightarrow n_y$, $p_h \rightarrow n_h \rightarrow n_i \rightarrow n_j$, and $p_h \rightarrow n_h \rightarrow n_i \rightarrow n_k$. (Here, $\leftrightarrow$ denotes a bidirectional edge, while $\rightarrow$ denotes a unidirectional one.) Because some of these paths also consume the capacity, we include the edges e_{n_i,n_h} and e_{n_h,n_i} to U_{p_j,p_h}. For any pair of adjacent triangular tiles in the routing network G, constraint (8) must be satisfied.

Another possible violation is shown in Fig. 6(b); if a flow goes along path $n_k \rightarrow n_i \rightarrow n_j$, the flow also passes through the space between pins p_j and p_h. However, this flow is not captured in constraint (8). To avoid such situation, we simply forbid this kind of flow by the following constraints:

$$f(e_{n_k,n_i}) + f(e_{n_j,n_i}) + f(e_{p_j,n_i}) = f(e_{n_i,n_h}), \tag{9}$$

$$f(e_{n_h,n_i}) = f(e_{n_i,n_k}) + f(e_{n_i,n_j}). \tag{10}$$

Equations (9) and (10) should be satisfied for each internal node in N. After combining with constraint (8), we can avoid the spacing violation between pins p_j and p_h. This is because we do not allow detoured paths inside a tile by (9) and (10). All legal paths are enumerated in constraint (8), and no other paths consume the capacity between pins p_j and p_h. As a result, we only need these constraints to avoid spacing violation in adjacent tiles.

Constraint (8) should be generated for any pair of adjacent triangular tiles in the routing network G, and constraints (9) and (10) are applied to all internal nodes. As these three additional constraints are added to the previous LP formulation described in Section IV-B, the wire spacing violation is avoided. Unfortunately, the unimodularity property of the constraints no longer holds. The variables in the optimal solution might be nonintegral. Thus, to ensure the amount of flow on every edge to be integral, we need to add the following constraint:

$$\forall f(e_{i,j}) \in Z_0^+. \tag{11}$$

In the ILP-based model, the sufficient condition is satisfied. That is, if a routing instance satisfies constraints (2)–(11), this instance is guaranteed to be routable on the staggered pin array without any design rule violation. However, the model [constraints (2)–(11)] could be over-constraining because some paths counted in constraint (8) may be redundant. In Fig. 6(a), for example, the redundant paths are as follows: $n_k \leftrightarrow n_i \leftrightarrow n_h \leftrightarrow n_x$ and $n_j \leftrightarrow n_i \leftrightarrow n_h \leftrightarrow n_y$. As a result, the necessary condition may not be satisfied.[1] Fig. 7 illustrates an example that our model is over-constraining. Given a staggered pin array with the configuration shown in Fig. 7, the capacity of each side of a tile is four, and the capacity between p_j and p_h is only six. Because the sum of the total flows shown in Fig. 7 is seven, constraint (8) is not satisfied, implying that our model cannot allow these legal flows in adjacent tiles. Therefore, our model gives only a sufficient condition and is not a necessary condition.

[1]The properties of our model will be further discussed in Section VI.

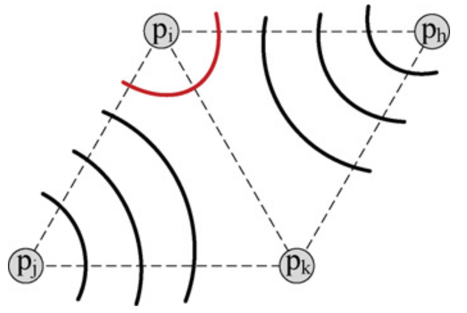


Fig. 7. Example shows that our model is over-constraining.

The ILP-based approach is often time-consuming. However, our problem size, the size of a staggered pin array, is much smaller than the size of full-chip routing. Experimental results show that our ILP approach has reasonable running time even for test cases larger than the industrial grid-pin arrays listed in [4]. According to the ILP formulation, we have the following theorem.

Theorem 2: If inequality (1) holds, a solution from our ILP formulation guarantees a routable instance on a staggered pin array, and the formulation has $O(|P|)$ variables and $O(|P|)$ constraints, where P is the set of all pins (nodes).

Proof: Based on the previous discussion in this section, we need to consider wire spacing violations between adjacent tiles. The four extra constraints [constraints (8)–(11)] are added to our ILP formulation to avoid spacing violations. Based on earlier discussions, we only need to consider these four constraints to avoid spacing violations in adjacent tiles because all legal paths are enumerated in constraint (8). Thus, we can use our ILP formulation to find legal escape routes without any wire spacing violation.

The ILP formulation is based on LP formulation and to add extra constraints (8)–(11). According to Theorem 1, the number of ILP variables also is $O(|P|)$, and the numbers of constraints (2)–(7) are $O(|P|)$. Besides, the numbers of constraints (8)–(10) are linear to the number of triangular models, which are linear to the number of pins; the number of constraint (11) is $O(|E|)$ and is linear to the number of pins according to the proof of Theorem 1. Therefore, the total number of ILP constraints is $O(|P|)$. ■

Now we illustrate the motivation for our triangular-tile model. For a triangular tile in a staggered pin array, the basic tile model is shown in Fig. 8(a). The model has one central node, three directed edges from each pin node to the central node, and three undirected edges among central nodes in the tile and its adjacent tiles. The capacity of each directed edge is one, and that of each undirected edge is equal to the capacity of the corresponding side in the tile. In addition, we set two types of capacity as follows: 1) the capacity of the central node in a tile, which represents the maximum number of wire segments

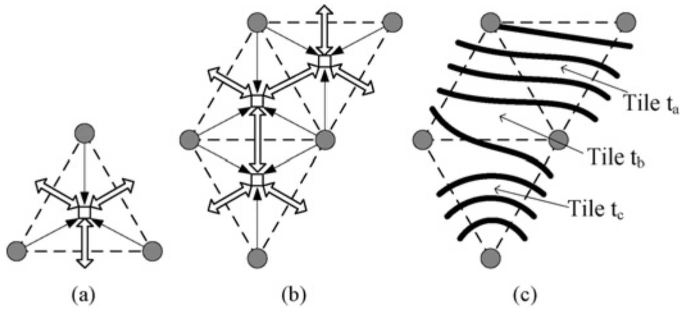


Fig. 8. (a) Basic triangular-tile model. (b) Partial routing network by connecting basic tile models. (c) Inequality (1) holds, but the basic tile model cannot capture the flows.

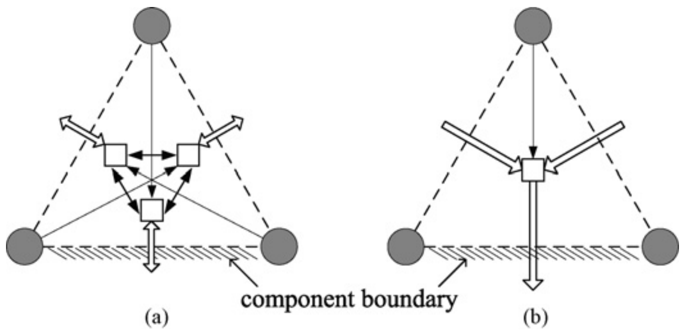


Fig. 9. Boundary tile model simplification. Each original boundary tile model in (a) can be simplified to (b) by reducing both the number of internal nodes and the number of edges.

that can be accommodated in the tile, and 2) the capacity of adjacent tiles, which is the maximum number of wire segments that can be accommodated in the adjacent tiles. Therefore, the total capacity of the two central nodes in adjacent tiles should be less or equal to the capacity of the adjacent tiles. Then we consider the partial routing network with two pairs of adjacent tiles as shown in Fig. 8(b). We assume that the capacity of each side and each central node is four, and inequality (1) holds. To avoid spacing violation, the capacity of adjacent tiles is seven rather than eight. However, the basic tile model loses some solution space. For example, the flows in Fig. 8(c) are valid because no spacing violation occurs in these tiles. Although the flows can correctly be captured in the tiles t_b and t_c, we cannot use the basic tile model to capture the flows in the tiles t_a and t_b. This is because the flow number in the tiles t_a and t_b is eight ($4+4=8$), but the capacity for adjacent tiles in the basic model is seven. Note that the situation does not occur in our proposed approach. Our formulation can correctly capture the flows in Fig. 8(c).

D. LP/ILP Reduction

To further reduce the running time of our LP/ILP-based approaches, we present a technique to reduce the number of variables and constraints in our LP/ILP formulation. We denote a routing tile with one side on a component boundary as a boundary tile. If a flow comes into a boundary tile, the best path of the flow is to directly go to the corresponding component boundary. This is because the capacity of the side on the component boundary is always larger or equal to the capacity of other sides in the tile. Any flow from the boundary tile to its adjacent tiles would induce redundant wire length. Therefore, the flow paths from the internal nodes of the boundary tile to those of the adjacent tiles are not required. We can simplify the boundary tile model as shown in Fig. 9. For each original boundary model in Fig. 9(a), we can simplify the tile model by reducing the number of the internal nodes from three to one and also removing corresponding edges. See Fig. 9(b) for the simplified boundary tile model.

We have the following theorem for the effectiveness of our reduction technique.

Theorem 3: Given an $m \times n$ staggered pin array, our reduction technique can reduce the number of nodes from $\lceil \frac{m}{2} \rceil n + \lfloor \frac{m}{2} \rfloor n + 5 \lfloor \frac{m}{2} \rfloor + 6mn - 9m - 6n + 9$ to $\lceil \frac{m}{2} \rceil n + \lfloor \frac{m}{2} \rfloor n + \lfloor \frac{m}{2} \rfloor + 6mn - 9m - 10n + 13$ (i.e., reduced by $4(\lfloor \frac{m}{2} \rfloor + n - 1)$).

Proof: For an $m \times n$ staggered pin array, the number of total pin nodes is $\lceil \frac{m}{2} \rceil n + \lfloor \frac{m}{2} \rfloor (n-1)$ and the number of total tiles is $(m-1)(2n-3)+2\lfloor \frac{m}{2} \rfloor$. Since each tile has three internal nodes without our reduction technique, the number of total internal nodes is $3\left((m-1)(2n-3)+2\lfloor \frac{m}{2} \rfloor\right)$. Therefore, the number of total nodes in the whole routing network without reduction is $\lceil \frac{m}{2} \rceil n + \lfloor \frac{m}{2} \rfloor n + 5\lfloor \frac{m}{2} \rfloor + 6mn - 9m - 6n + 9$. After our reduction technique, the number of nodes in a boundary tile can be reduced from three to one. Further, the number of boundary tiles is $2(\lfloor \frac{m}{2} \rfloor + n - 1)$, and thus the total number of reduced nodes is $4(\lfloor \frac{m}{2} \rfloor + n - 1)$. As a result, the number of nodes can be reduced from $\lceil \frac{m}{2} \rceil n + \lfloor \frac{m}{2} \rfloor n + 5\lfloor \frac{m}{2} \rfloor + 6mn - 9m - 6n + 9$ to $\lceil \frac{m}{2} \rceil n + \lfloor \frac{m}{2} \rfloor n + \lfloor \frac{m}{2} \rfloor + 6mn - 9m - 10n + 13$. ■

E. Noncrossing Theorem

Although the edges in the triangular-tile model shown in Fig. 5(a) have crossings, the global routing solution obtained by our LP/ILP approach will not have any crossing.

Theorem 4: The global routing solution obtained by our LP/ILP approach has no crossings.

Proof: According to our tile model in Fig. 5(a), flow crossings only occur by the following two situations: 1) flows from pins to internal nodes and flows between internal nodes, and 2) flows from pins to internal nodes and other flows from pins to internal nodes. However, the two types of flow crossings always induce longer wire length in the tile and its adjacent tiles. For each pair of flows which induce a crossing, we can always move the crossing flows to proper edges in the tile and its adjacent tiles to remove the crossing, and thus the new wire length will be shorter. For the first situation, we assume that the crossing flows are in e_{n_k,n_j} and e_{p_i,n_i} as shown in Fig. 5(a). The flow in e_{p_i,n_i} can be moved to the edge from p_i in the adjacent tile, which is nearby the node n_j; we then move the one unit flow in e_{n_k,n_j} to e_{n_k,n_i} in the tile. For the second situation, we assume that the crossing flows are in e_{p_i,n_i} and e_{p_j,n_j}. The flows in e_{p_i,n_i} and e_{p_j,n_j} can be moved to other edges also from p_i and p_j in other tiles surrounded with p_i and p_j, respectively. In addition, if we have a directed edge from a pin p_x, the pin p_x must be located inside a staggered pin array according to our routing network construction (because pins

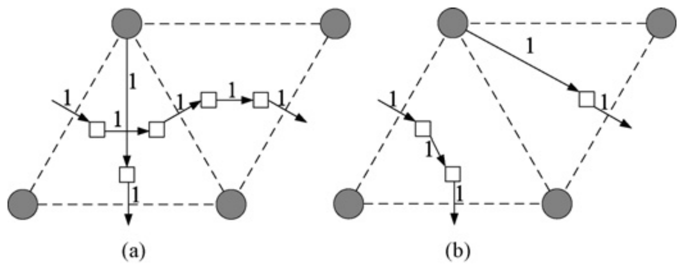


Fig. 10. If the solution contains two crossing flows in (a), it can be replaced by (b) to get a better solution.

at a array boundary do not need to escape). Flows in an edge crossing with the directed edge from the pin p_x must keep going to an adjacent tile rather than end in the current tile, and thus we can find one edge from the pin p_x in the adjacent tile. As a result, we always have proper edges in some adjacent tiles and can move flows to proper edges. The move procedure can be categorized into two cases: 1) the move finally reaches a boundary of the staggered pin array, and 2) the move can return to the original tile where we initialized the move. For the first case, the move finally reaches a boundary tile with one side on a component boundary. To obtain smaller wire length, flows in a boundary tile should directly go to the corresponding component boundary, which has been discussed in Section IV-D. Therefore, there is no crossing in the boundary tile, and the crossing removal procedure is finished. The second case implies that we should move another crossing flow to change the location of the crossing. Because all escaped pins must finally be routed to an array boundary, we can move another flow to make a crossing toward a boundary tile gradually. Similar to the first case, therefore, the crossing finally exists in its corresponding boundary tile and can be eliminated. For example, suppose we are given an optimal solution with crossing flows, it must contain at least one pair of adjacent triangular tiles with two crossing flows of one unit, as shown in Fig. 10(a). However, we can replace these two flows in Fig. 10(a) with the two flows in Fig. 10(b), thus achieving shorter total wire length while maintaining the same total flow amount. In this case, our approach will choose the flows in Fig. 10(b) as the optimal solution, but not those in Fig. 10(a). Please note that the input/output flows on the boundaries of these two tiles are exactly the same in Fig. 10(a) and (b). As a result, the new solution after replacing these two unit flows is still valid, and is better than the original one in terms of wire length, so the original one cannot be an optimal solution. This theorem is then proved by contradiction. Therefore, there exist no wire crossings in the global routing solution obtained by our LP/ILP approach. ■

F. Detailed Routing

In detailed routing, we perform the orthogonal-side wire routing based on the result of global routing. The number of wires in a tile is derived from $f(e_i, j)$ in the global routing result. Since we intend to maximize the number of the wires passing through the spacing between adjacent pins, we let each wire in a tile be orthogonal to the side of the tile, as we discussed in Section II-B. Fig. 11 shows an example for our detailed routing. After the global routing, we obtain the flow of each edge in the tile model, illustrated in Fig. 11(a), so we know that there are two wires from the left side of the tile to the right side of the tile and one wire from the bottom side to the right side, as shown in Fig. 11(b). Then, we can determine the positions of wires and boundary intersections. Finally, we route the wires and let them be orthogonal to the side of the tile; see Fig. 11(c) for the resulting routing configuration.

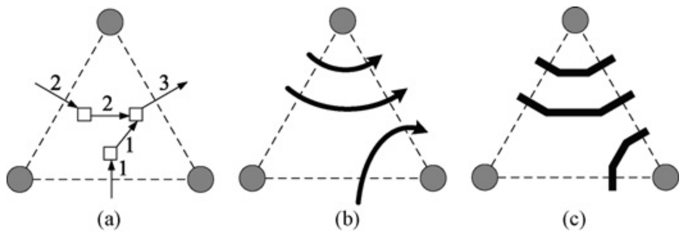


Fig. 11. Example of detailed routing.

TABLE I
BENCHMARK CIRCUITS FOR ESCAPE ROUTING

Circuits	Size	#Escape Pin	Capacity
case 1	15 × 15	48	1
case 2	23 × 17	130	2
case 3	17 × 35	280	3
case 4	35 × 35	252	2
case 5	65 × 65	705	3
case 6	35 × 35	600	5
case 7	65 × 65	1108	5
case 8	47 × 23	582	5
case 9	21 × 37	496	5
case 10	85 × 85	1407	5

V. Experimental Results

We implemented our algorithm in the C++ programming language and performed experiments on a 2.6 GHz AMD Opteron Linux workstation with 6 GB memory. We used the CPLEX [24] library to solve the LP and ILP problems.

To validate the effectiveness of our approach, we created ten staggered test cases based on the parameters of the grid-pin arrays listed in [4], including the array sizes, pin numbers, and capacity. We also randomly generated some test cases with sizes bigger than the test cases in [4] to evaluate the efficiency of our approach, especially for our ILP formulation.

The test cases can be classified into two categories according to whether inequality (1) holds.

1) In case1 to case5, inequality (1) does not hold, so we can adopt our LP approach to route these cases.
2) In case6 to case 10, inequality (1) holds, which means that wires passing through two adjacent tiles may have wire spacing violation. The LP formulation is not applicable for these cases, and thus the ILP formulation approach is adopted to route them.

Table I shows the statistics of these test cases. "Size" represents the size of the staggered pin array in $m \times n$ format, which is defined in Section II-A; "#Escape Pin," the number of assigned pins for escape routing; and "Capacity", the number of nets that can pass through two adjacent pins without violating design rules.

To demonstrate the effectiveness of our proposed orthogonal-side wiring style, we also implemented a

TABLE II
OUR APPROACH BY THE LP FORMULATION

Circuits	90°	Ours w/o Reduction		Ours w/ Reduction	
		Wire length (μm)	CPU times (s)	Wire length (μm)	CPU times (s)
case 1	NA	14 707	0.19	14707	0.15
case 2	NA	68 948	0.46	68948	0.40
case 3	NA	206 879	0.63	206879	0.54
case 4	NA	238 731	1.75	238731	1.52
case 5	NA	1 549 776	16.36	1 549 776	15.53
Comp.		1.00	1.16	1.00	1.00

TABLE III
OUR APPROACH WITH THE ILP FORMULATION

Circuits	90°	LP	Ours w/o Reduction		Ours w/ Reduction	
			Wire length (μm)	CPU times (s)	Wire length (μm)	CPU times (s)
case 6	NA	NA	1 048 471	3.64	1 048 471	3.16
case 7	NA	NA	3 738 714	36.41	3 738 714	33.51
case 8	NA	NA	836 276	2.88	836 276	2.59
case 9	NA	NA	622 314	1.70	622 314	1.38
case 10	NA	NA	5 796 152	95.21	5 796 152	89.28
Comp.			1.00	1.13	1.00	1.00

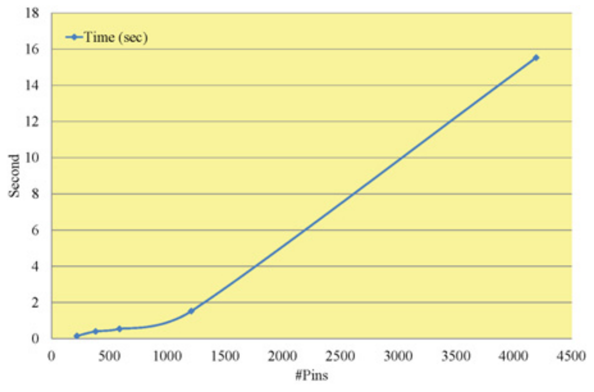


Fig. 12. CPU times for our LP formulation.

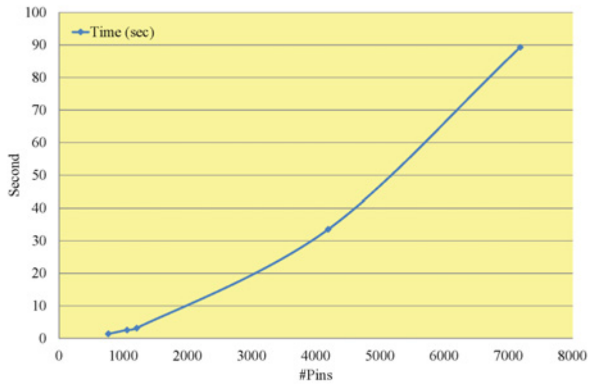


Fig. 13. CPU times for our ILP formulation.

90° routing algorithm, which formulates the capacity of a triangular tile as the maximum number of vertical/horizontal wires that can pass through the tile. In this 90° routing formulation, the capacity of the tile in Fig. 2(a) would be two.

Table II shows the results of our LP approach for the first five test cases. All cases are successfully routed in reasonable time. As for the traditional 90° routing algorithm, since the capacity of tiles is under-estimated, the solution space is greatly lost. As a result, the 90° routing algorithm failed to complete the routing for all the cases, denoted by "NA" (not available).

Table III shows the results of our ILP approach for the latter five test cases. Similarly, the traditional 90° routing algorithm cannot complete routing for any case. And the results of the LP approach for these test cases have wire spacing violation in adjacent tiles, as we discussed in Section II. Consequently, the results are denoted as NA. Only our ILP approach is able to produce results with 100% routability. The experimental results show the effectiveness of our approaches.

We also implemented our reduction technique to further reduce the CPU times. As described in Section IV-D, for an

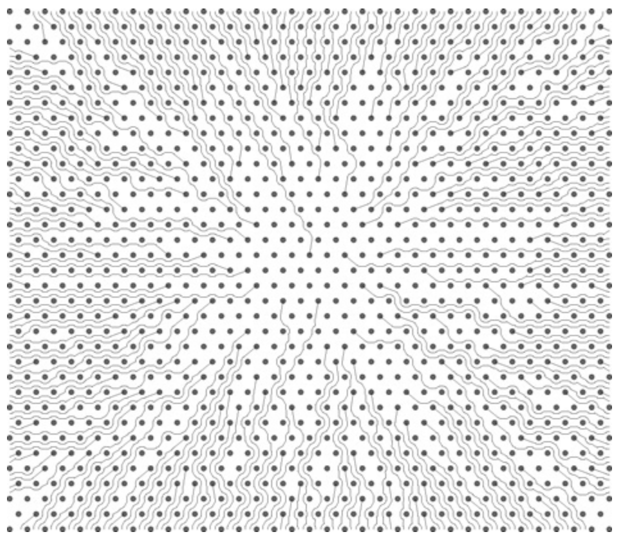


Fig. 14. Resulting layout of case 4 using our LP formulation.

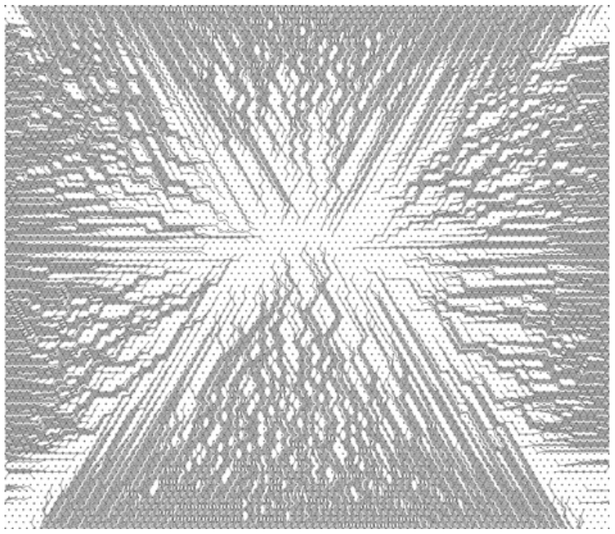


Fig. 15. Resulting layout of case 10 using our ILP formulation.

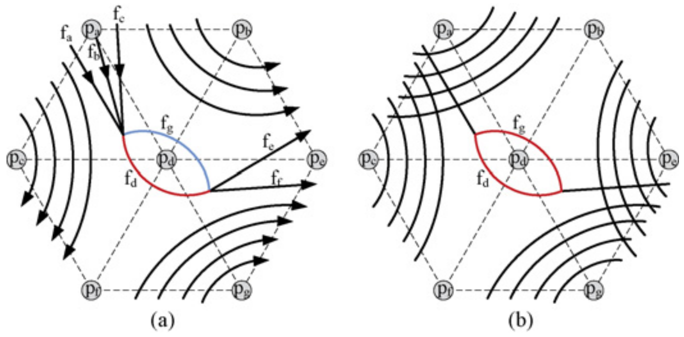


Fig. 16. Routing result in our routing network. (a) Flow f_d can be replaced by the flow f_g to get an equal or smaller length in our routing network. (b) Case cannot happen because the numbers of flows of the specified sides exceed the capacity.

$m \times n$ staggered pin array, the total reduced node number of our routing network is $4(\lfloor \frac{m}{2} \rfloor + n - 1)$. The results of our LP and ILP approaches with the reduction technique are also shown in Table II and III, respectively. The reduction technique reduces the average CPU time by 16% for the LP formulation and 13% for the ILP formulation, with no wire length overhead.

In Fig. 12, the CPU time of our LP formulation with reduction is plotted as a function of the number of input pins. Empirically, the CPU time of our LP approach with reduction is between linear and quadratic (about $P^{1.54}$) to the number of input pins (P), with the least-square analysis for the log–log plot of the function. Our ILP approach has a similar result. In Fig. 13, the CPU time of our ILP approach with reduction is plotted as a function of the number of input pins. The empirical time complexity of our ILP approach with reduction is between linear and quadratic (about $P^{1.87}$) to the number of input pins (P), again with the least-square analysis for the log–log plot of the function, which is slightly higher than that of the LP approach.

Fig. 14 shows the routing result of case 4 generated by our LP formulation, and Fig. 15 illustrates the routing result of case 10 generated by our ILP formulation.

VI. Conclusion

In this paper, we proposed an escape routing algorithm for the staggered pin array. The traditional escape routing approaches for gird pin arrays cannot fully utilize the routing resource of the staggered pin array. Based on our analysis of a triangular tile in the staggered array, we designed a triangular-tile model that fully utilized the routing capacity. We also developed an LP/ILP formulation-based global routing algorithm for the problem, and a track-assignment-based detailed routing algorithm. Experimental results showed that our proposed algorithm is much more effective than the traditional 90° routing approach.

As discussed earlier, our ILP-based model is only a sufficient condition and not a necessary one. Although our ILP formulation could be over-constraining with constraint (8) for the case with two adjacent tiles, we conjecture that the formulation can still provide a necessary condition for the case with the whole routing network. Fig. 16 shows an example. Assume that the capacity of a tile side is five, and the capacity between adjacent tiles is eight. We define tile $p_a p_b p_c$ to be the tile formed by the three adjacent pins p_a, p_b, and p_c. In Fig. 16(a), if we have the flow f_d, which is from flow f_a, f_b, or f_c, and to flow f_e or f_f, constraint (8) is violated in the adjacent tiles $p_c p_d p_f$ and $p_d p_f p_g$ because the number of flows is nine. We cannot allow these flows in the two adjacent tiles $p_c p_d p_f$ and $p_d p_f p_g$. However, the flow f_d can be replaced by the flow f_g to get an equal or smaller length in our routing network. Our model can get the routing result because constraint (8) is not violated in all adjacent tiles. Note that the numbers of flows in the adjacent tiles $p_a p_c p_d$ and $p_a p_b p_d$, and the adjacent tiles $p_b p_d p_e$ and $p_d p_e p_g$ are not changed after we replace the flow f_d by the flow f_g. In Fig. 16(b), if we have the flow f_d, constraint (8) is violated in the adjacent tiles $p_c p_d p_f$ and $p_d p_f p_g$. On the other hand, if we have the flow f_e, constraint (8) is violated in the adjacent tiles $p_a p_b p_d$ and $p_b p_d p_e$. It seems that we cannot replace the flow f_d by the flow f_g in our routing network. However, this situation cannot happen because the number of flows between the pins p_a and p_c and the number of flows between the pins p_e and p_g exceed the capacity. Therefore, we can replace the flow f_d by the flow f_g to get the correct number of flows in our routing network, even though our model is over-constraining. Our future work lies in the proof of this conjecture.

References

[1] Y.-K. Ho, H.-C. Lee, and Y.-W. Chang, "Escape routing for staggered-pin-array PCBs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, 2011, pp. 306–309.

[2] T. Yan and M. D.-F. Wong, "Recent research development in PCB layout," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, 2010, pp. 398–403.

[3] J.-W. Fang, I.-J. Lin, Y.-W. Chang, and J.-H. Wang, "A network-flow based RDL routing algorithm for flip-chip design," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 8, pp. 1417–1429, Aug. 2007.

[4] T. Yan and M. D.-F. Wong, "A correct network flow model for escape routing," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jul. 2009, pp. 332–335.

[5] J.-W. Fang, K.-H. Ho, and Y.-W. Chang, "Routing for chip-package-board co-design considering differential pairs," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, Nov. 2008, pp. 512–517.

[6] J.-W. Fang, M. D.-F. Wong, and Y.-W. Chang, "Flip-chip routing with unified area-I/O pad assignments for package-board co-design," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jul. 2009, pp. 336–339.

[7] J.-W. Fang and Y.-W. Chang, "Area-I/O flip-chip routing for chip-package co-design considering signal skews," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 5, pp. 711–721, May 2010.

[8] X. Liu, Y. Zhang, G. K. Yeap, C. Chu, J. Sun, and X. Zeng, "Global routing and track assignment for flip-chip designs," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jun. 2010, pp. 90–93.

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

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

[11] R. Wang, R. Shi, and C.-K. Cheng, "Layer minimization of escape routing in area array packaging," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Des.*, Nov. 2006, pp. 815–819.

[12] L. Luo and M. D. F. Wong, "Ordered escape routing based on Boolean satisfiability," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Mar. 2008, pp. 244–249.

[13] L. Luo and M. D. F. Wong, "On using SAT to ordered escape problems," in *Proc. IEEE/ACM Asia South Pacific Des. Autom. Conf.*, Jan. 2009, pp. 594–599.

[14] L. Luo, T. Yan, Q. Ma, M. D. F. Wong, and T. Shibuya, "B-escape: A simultaneous escape routing algorithm based on boundary routing," in *Proc. Int. Symp. Phys. Des.*, 2010, pp. 19–25.

[15] M. M. Ozdal and M. D. F. Wong, "Algorithms for simultaneous escape routing and layer assignment of dense PCBs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 25, no. 8, pp. 1510–1522, Aug. 2006.

[16] M. M. Ozdal, M. D. F. Wong, and P. S. Honsinger, "Simultaneous escape-routing algorithms for via minimization of high-speed boards," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 27, no. 1, pp. 84–95, Jan. 2008.

[17] N. M. Gasparini and B. K. Bhattacharyya, "A method of designing a group of bumps for C4 packages to maximize the number of bumps and minimize the number of package layers," in *Proc. 44th Electronic Components Technol. Conf.*, 1994, pp. 695–699.

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

[19] *Intel Pentium Processor Family Developer's Manual*, Intel, Santa Clara, CA, USA, 1997.

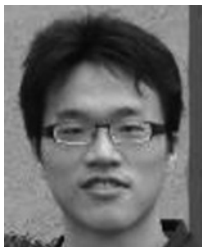
[20] R. Shi and C.-K. Cheng, "Efficient escape routing for hexagonal array of high density I/Os," in *Proc. ACM/IEEE Des. Autom. Conf.*, Jul. 2006, pp. 1003–1008.

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

[22] M. Yannakakis, "On a class of totally unimodular matrices," in *Proc. 21st Annu.Symp. Foundations Comput. Sci.*, 1980, pp. 10–16.

[23] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica,* vol. 4, no. 4, pp. 373–395, 1984.

[24] IBM ILOG CPLEX Optimizer [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/



Yuan-Kai Ho received the B.S. and M.S. degrees in electronic engineering from Chung Yuan Christian University, Taoyuan, Taiwan, in 2006 and 2008, respectively. He is currently pursuing the Ph.D. degree through the Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan.

His current research interests include PCB routing and flip-chip routing.

Hsu-Chieh Lee received the B.S. and M.S. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2008 and 2011, respectively.

He is currently an employee of Google Inc., Mountain View, CA, USA. His current research interests include PCB routing, flip-chip routing, and cross-domain co-design integration.

Yao-Wen Chang (S'94–A'96–M'96–SM'12–F'13) received the B.S. degree from National Taiwan University (NTU), Taipei, Taiwan, in 1988, and the M.S. and Ph.D. degrees from the University of Texas at Austin, Austin, TX, USA, in 1993 and 1996, respectively, all in computer science.

He is currently an Associate Dean of the College of Electrical Engineering and Computer Science, the Director of the Graduate Institute of Electronics Engineering, and a Distinguished Professor in the Department of Electrical Engineering, NTU. He has been working closely with the industries related to manufacturability and very large scale integration (VLSI) physical design. He has co-edited one textbook on electronic design automation and co-authored one book on routing, as well as more than 220 ACM/IEEE conference and journal papers in these areas. His current research interests include VLSI physical design and design for manufacturability.

Dr. Chang was a First Place winner of four recent contests: 1) the 2012 ACM/IEEE Design Automation Conference (DAC) Placement Contest, 2) the 2012 ACM International Symposium on Physical Design (ISPD) Discrete Gate Sizing Contest, 3) the 2011 IEEE Council on Electronic Design Automation (CEDA) International Workshop on Power and Timing Modeling, Optimization and Simulation Timing Analysis Contest, and 4) the 2009 ACM ISPD Clock Tree Synthesis Contest. He is a five-time winner of the ACM ISPD contests on placement, global routing, clock network synthesis, and discrete gate sizing, and was the recipient of six Best Paper Awards [the IEEE International Conference on Computer Design (ICCD), etc.] and 21 Best Paper Award nominations from DAC (five times), the International Conference on Computer-Aided Design (ICCAD) (four times), ISPD (six times), etc., over the past ten years. He has received many research and teaching awards, such as the Distinguished Research Award (highest honor) from the National Science Council of Taiwan (twice), the IBM Faculty Award (three times), Chinese Institute of Electrical Engineering Distinguished Electrical Engineering Professorship, Macronix International Corporation Young Chair Professorship, and Excellent Teaching Award from NTU (seven times). He is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and an Editor of the IEEE DESIGN AND TEST OF COMPUTERS. He has served as the General and Steering Committee Chairs of ISPD, and the Program Chairs of ASP-DAC, the International Conference on Field Programmable Technology (FPT), ICCAD, and ISPD. He is on the IEEE CEDA Executive Committee, the ICCAD Executive Committee, the ASP-DAC Steering Committee, and the ACM/SIGDA Physical Design Technical Committee. He has served on technical program committees of major electronic design automation (EDA) conferences, including ASP-DAC, DAC, Design, Automation and Test in Europe, the International Conference on Field Programmable Logic, FPT, Great Lakes Symposium on VLSI, ICCAD, ICCD, ISPD, Workshop on System Level Interconnect Prediction, the IEEE System-on-Chip Conference, and VLSI-DAT. He has served as an independent Board Director of Genesys Logic, a Technical Consultant of Faraday, MediaTek, and RealTek, a Chair of the EDA Consortium of the Ministry of Education, Taiwan, and a member of the Board of Governors of the Taiwan IC Design Society.